Fall 12-31-2018

Load balancing and scalable clos-network packet switches

Oladele Theophilus Sule

New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

LOAD BALANCING AND SCALABLE CLOS-NETWORK PACKET SWITCHES

by
Oladele Theophilus Sule

In this dissertation three load-balancing Clos-network packet switches that attain 100% throughput and forward cells in sequence are introduced. The configuration schemes and the in-sequence forwarding mechanisms devised for these switches are also introduced. Also proposed is the use of matrix analysis as a tool for throughput analysis.

In Chapter 2, a configuration scheme for a load-balancing Clos-network packet switch that has split central modules and buffers in between the split modules is introduced. This switch is called split-central-buffered Load-Balancing Clos-network (LBC) switch and it is cell based. The switch has four stages, namely input, central-input, central-output, and output stages. The proposed configuration scheme uses a pre-determined and periodic interconnection pattern in the input and split central modules to load-balance and route traffic. The LBC switch has low configuration complexity. The operation of the switch includes a mechanism applied at input and split-central modules to forward cells in sequence. The switch achieves 100% throughput under uniform and nonuniform admissible traffic with independent and identical distributions (i.i.d.). The high switching performance and low complexity of the switch are achieved while performing in-sequence forwarding and without resorting to memory speedup or central-stage expansion. This discussion includes both throughput analysis, where the operations that the configuration mechanism performs on the traffic traversing the switch are described, and a proof of in-sequence forwarding. Simulation analysis is presented as a practical demonstration of the switch performance on uniform and nonuniform i.i.d. traffic.

In Chapter 3, a three-stage load balancing packet switch and its configuration scheme are introduced. The input- and central-stage switches are bufferless crossbars and the output-stage switches are buffered crossbars. This switch is called ThRee-stage Clos-network swItch and has queues at the middle stage and DEtermiNisTic scheduling (TRIDENT) and it is cell based. The proposed configuration scheme uses a pre-determined and periodic interconnection pattern in the input and central modules to load-balance and route traffic; therefore, it has low configuration complexity. The operation of the switch includes a mechanism applied at input and output modules to forward cells in sequence.

In Chapter 4, a highly scalable load balancing three-stage Clos-network switch with Virtual Input-module output queues at ceNtral stagE (VINE) and crosspoint-buffers at output modules and its configuration scheme are introduced. VINE uses space switching in the first stage and buffered crossbars in the second and third stages. The proposed configuration scheme uses pre-determined and periodic interconnection patterns in the input modules for load balancing. The mechanism applied at the inputs, used to forward cells in sequence, is also introduced. VINE achieves 100% throughput under uniform and nonuniform admissible i.i.d. traffic. VINE achieves high switching performance, low configuration complexity, and in-sequence forwarding without resorting to memory speedup.

In Chapter 5, matrix analysis is introduced as a tool for modeling, describing the internal operations, and analyzing the throughput of a packet switch.

LOAD BALANCING FOR PACKET SWITCHES AND DATANCETER NETWORKS

by
Oladele Theophilus Sule

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

December 2018

APPROVAL PAGE

LOAD BALANCING AND SCALABLE CLOS-NETWORK PACKET SWITCHES

Oladele Theophilus Sule

Prof. Roberto Rojas-Cessa, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Prof. Nirwan Ansari, Committee Member — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Prof. John D. Carpinelli, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Prof. Ziqian Dong, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, New York Institute of Technology

---

Prof. Qing Liu, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Oladele Theophilus Sule

Degree: Doctor of Philosophy

Date: December 2018

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2018
- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2012
- Bachelor of Science in Physics,
  Benson Idahosa University, Benin City, Nigeria, 2008

Major: Electrical Engineering

Presentations and Publications:

O. T. Sule, R. Rojas-Cessa, Z. Dong, and C.-B. Lin, "A Split-Central-Buffered Load-Balancing Clos-Network Switch with In-Order Forwarding," Accepted in IEEE/ACM Transactions on Networking. IEEE DOI 10.1109/TNET.2018.2883747.

O. T. Sule and R. Rojas-Cessa, "TRIDENT: A Load-Balancing Clos-network Packet Switch with Queues between Input and Central Stages and In-Order Forwarding," Under Review.

O. T. Sule and R. Rojas-Cessa, "VINE: Load-Balancing Clos-Network Switch with Virtual Input-Module Output Queues at Central Stage," Under Review.

O. T. Sule and R. Rojas-Cessa, "Modeling of Network Packet Switches Using Matrix Analysis," Under Review.

ACKNOWLEDGMENT

Firstly, I give all the thanks, glory, and adoration to God the Father, God the Son, and God the Holy Spirit, for the wisdom, knowledge, understanding, and perseverance throughout the PhD program.

I would like to thank my advisor, Professor Roberto Rojas-Cessa for his guidance and advise. I would also like to thank my committee members: Professor Nirwan Ansari, Professor John D. Carpinelli, Professor Ziqian Dong, and Professor Qing Liu for their comments and suggestions.

Next, I would like to thank the staff at the Office of Graduate Studies at the New Jersey Institute of Technology and the staff at the Department of Electrical and Computer Engineering.

Finally, I would like to thank my mother, Margaret Asibor, my friends, and family for all their love and support.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

Clos-network switches are attractive for building large-size switches [19]. These switches mostly employ three stages, where each stage uses switch modules as building blocks. Each module is a small- or medium-size switch. Modules of the first, second, and third stages are often called input, central, and output modules, and they are denoted as IM, CM, and OM, respectively. Overall, Clos-network switches require fewer crosspoint elements, each of which is the atomic switching unit of a packet switch, than a single-stage switch of equivalent size, and thus they may require less building hardware. This trait of a Clos network often comes at the cost of an increased configuration complexity. In general, a Clos-network switch requires the configuration of the modules in every stage before packets are forwarded through. Moreover, owing to the multi-stage architecture of such switch, the time for switch reconfiguration increases as the number of stages holding dependences increases. In a multi-stage switch, there is a dependence when the configuration of a module is affected by the configuration of another. The required configuration time dictates the internal data transmission time, which in turn defines the minimum size of the internal data unit. Therefore, the configuration time of such switch must be kept to the shortest possible for a fast and efficient reconfiguration.

In the remainder of this dissertation, we consider the proposed packet switches to be cell based; this is, upon arrival in an input port of a switch, packets of variable size are segmented into fixed-size cells. Cells are forwarded through the switch to their destined outputs. Packets are re-assembled at the outputs of the switch.

Based on whether a stage performs space- (S) or memory-based (M) switching, Clos-network switches can be categorized into SSS (or S^3) [12,44,47,60,81], MSM [11,

16, 28, 33, 40, 41, 73, 80], MMM [13, 17, 23, 82], SMM [42, 46, 50, 78, 83], and SSM [48, 70, 71], among the most popular ones. From those, S^3 switches require small amounts of hardware but their configuration has been proven challenging as input-to-output path setup must be resolved before cells are transmitted. On the other hand, inclusion of memory in modules may relax the configuration complexity. However, configuration complexity has remained high despite using memory in every switch module because of internal blocking and the multiplicity of input-output paths associated with diverse queuing delays [13, 24]. Specifically, switches with buffered central or output stages are prone to forwarding packets out of sequence, making re-sequencing or in-sequence transmission mechanisms an added feature. Moreover, the number and size of queues in a module are restricted to the available on-chip real estate. This restriction plus the adopted in-sequence measures may exacerbate internal blocking that, in turn, may lead to performance degradation [23].

Minimizing the complexity of the central module of a Clos-network switch has been of research interest in recent years. Hassen et. al proposed a Clos-network switch that combines different switching stages [29]. In this work, central modules are replaced with multi-directional networks-on-chip (MDN) modules. The switch uses a static dispatching scheme from the input/output modules, for which every input constantly delivers packets to the same MDN module, and adopts inter-central-module routing to enable forwarding the cells to the final destination. However, this switch may forward cells to output port out of sequence if cells from the same flow are routed through different paths on the central modules.

Load balancing traffic prior to routing it towards the destined output is a technique that not only improves switching performance but also reduces the configuration complexity of a packet switch when the load-balancing and routing follows a deterministic schedule [9]. Such a schedule may be obtained as an application

of matrix decomposition [5,43]. This technique enables high performance not only on switches but also on a large number of network applications [22, 35, 76].

A switch that load-balances traffic may need at least two stages to operate; one for load balancing and the other for routing cells to their destined outputs [9]. A switch with such a deterministic and periodic schedule may require the use of queues between the load-balancing and routing stages. However, placing such queues and enabling multiple interconnection paths between an input and an output make load-balancing switches susceptible to forwarding cells out of sequence [9]. This issue has been addressed by introducing either re-sequencing buffers at the output ports [10] or mechanisms that prevent out-of-sequence forwarding [39, 64]. However, these approaches are either complex or degrade switching performance.

Load balancing has been applied to Clos-network switches [13,84]. For example, Zhang et al. [84] proposed an SMM switch that adopts the two-stage load-balanced Birkhoff-von Neumann switch in each central module but has no input port buffers. Here, a central module consists of two $k \times k$ bufferless crossbar switches and k buffers in between the crossbars. The switch performs load balancing at the input module and the first stage of the load-balanced Birkhoff-von Neumann switch. Each of these queues accommodates up to one cell to guarantee the transmission of cells in sequence. However, the distance between modules in a large switch requires larger queue sizes for which this switch would suffer from out-of-sequence forwarding.

The switches discussed above, either suffer from limited switching performance, high complexity, or out-of-sequence forwarding. These drawbacks then raise the question, can a load-balancing Clos-network switch achieve high switching performance, low configuration complexity, and in-sequence cell forwarding without resorting to memory speedup?

In this dissertation, we aim at answering this question by proposing three load-balancing Clos-network switches: A split-central-buffered Load-Balancing Clos-

network (LBC) switch, ThRee-stage Clos-network swItch with queues at the middle stage and DEtermiNisTic scheduling (TRIDENT), and VINE: Load-Balancing Clos-Network Switch with Virtual Input-Module Output Queues at Central Stage.

CHAPTER 2

A SPLIT-CENTRAL-BUFFERED LOAD-BALANCING CLOS-NETWORK SWITCH WITH IN-ORDER FORWARDING

2.1 Introduction

In this chapter we propose a split-central-buffered Load-Balancing Clos-network (LBC) switch. The switch has a split central module and queues in between. The switch employs predetermined and periodic interconnection patterns to interconnect the inputs and outputs of the switch modules. The switch load balances the incoming traffic and switches the cells towards the destined outputs, both with minimum configuration complexity. The result is a switch that attains high throughput under admissible traffic with independent and identical distribution (i.i.d.) and uses a configuration scheme with $O(1)$ complexity. The switch also adopts an in-sequence forwarding mechanism at the input queues to keep cells in sequence despite the presence of buffers between the split CMs.

In summary, the contributions of this chapter are: 1) the proposal of a configuration scheme for a split-central-buffered load-balancing switch such that the attained throughput is 100% under admissible traffic while having $O(1)$ scheduling complexity, 2) the proposal of an in-sequence mechanism for forwarding of cells in sequence throughout the switch, 3) presentation of throughput analysis of the LBC switch for each of the stages that shows that the switch achieves 100% throughput under i.i.d. admissible traffic, and proof of the in-sequence capability of the proposed in-sequence forwarding mechanism.

The remainder of this chapter is organized as follows: In Section 2.2, we introduce and describe the LBC switch. In Section 2.3, we analyze the throughput performance of LBC. In Section 2.4, we analyze the in-sequence forwarding property

of the LBC switch. In Section 2.5, we present a simulation study on the performance of LBC. In Section 2.6, we present our conclusions.

2.2 Switch Architecture

The LBC switch has N inputs and N outputs, each denoted as $IP(i, s)$ and $OP(j, d)$, respectively, where $0 \leq i,\ j \leq k-1,\ 0 \leq s,\ d \leq n-1$, and $N = nk$. Figure 2.1 shows the architecture of the LBC switch. This switch has k $n \times m$ IMs and k $m \times n$ OMs. Each central module is split into two modules called central-input and -output modules, denoted as CIMs and COMs, respectively. The switch has m CIMs and the same number of COMs. Each CIM and COM is a $k \times k$ switch. In the remainder of this chapter, we set $n = k = m$ for symmetry and cost-effectiveness. The IMs, CIMs, and COMs are bufferless crossbars while the OMs are buffered ones.

The use of a split central module on this switch enables preserving staggered symmetry and in-order delivery [31] by using a pre-determined configuration in the IMs, CIMs, and COMs with a mirror sequence between CIMs and COMs. The staggered symmetry and in-order delivery refers to the fact that at time slot t, $IP(i, s)$ is connected to $COM(r)$, which in turn is connected to $OM(j)$. In the next time slot $(t+1)$, $IP(i, s)$ is connected to $COM((r+1) \mod m)$, which is connected to $OM(j)$. This property enables the configuration of IMs/CIMs and COMs to be easily represented with a pre-determined compound permutation that repeats every k time slots. This property also ensures that cells experience the same amount of delay for uniform traffic and the incorporation of a simple in-sequence mechanism. A switch with queues between IMs and CMs but without a split central module may require more complex load balancing and routing configurations to achieve the same objective.

Each input port has N virtual output queues (VOQs), denoted as $VOQ(i, s, j, d)$, to store cells destined to output port d at $OM(j)$. The combination of IMs and CIMs

form a compound stage, called the IM-CIM stage. The COMs and OMs operate as single stages. There are queues placed between CIMs and COMs to store cells coming from an IM and destined to OMs. These central queues may be implemented as virtual output port queues (VOPQs), as shown in Figure 2.2(a). Each VOPQ, denoted as $VOPQ(r, p, j, d)$, stores cells coming for $OP(j, d)$ through $L_{CIM}(r, p)$. As an alternative, to reduce the number of VOPQs for a large switch, we consider the use of virtual output module queues (VOMQs) instead, as shown in Figure 2.2(b). A VOMQ, denoted as $VOMQ(r, p, j)$, stores cells for all OPs at $OM(j)$. Each of these queues stores cells coming from $L_{CIM}(r, p)$ and destined to $OM(j)$. Compared to VOPQs, VOMQs introduce the possibility of cells experiencing head-of-line (HoL) blocking. However, as we show in Section 2.2.6, such HoL is not a concern when the switch is loaded with admissible traffic. The remainder of this chapter considers VOMQs, as this option stresses the load-balancing feature of LBC.

Every CIM has k L_{CIM} ports. Every $L_{CIM}(r, p)$ of a CIM is connected to one input $I_C(r, p)$ of the corresponding COM. The LCIM includes a set of k VOMQs, one per OM. Each OP has m crosspoint buffers, each denoted as $CB(r, j, d)$. A flow control mechanism operates between VOMQs and VOQs, and between CBs and VOMQs to avoid buffer overflow and this is described in Section 2.2.5. The VOMQs are off-chip. The switch has N LCIMs, and therefore N sets of k VOMQs each. Table 2.1 lists the notations used in the description of the LBC switch.

The following is a walk-through description of how the switch operates: After arriving at the IP, a cell is placed at the VOQ corresponding to its destination OP. The IP arbiter selects a VOQ to be served in a round-robin manner. When a VOQ is selected, the HoL cell is forwarded to a VOMQ at the LCIM identified by the current configuration of the IM and CIM. The VOMQ is the one associated with the OM that includes the destination OP of the cell. When the configuration of the COM permits forwarding to the destination OM, the cell is forwarded to the OM and stored at

the crosspoint buffer (CB) allocated for cells from the source COM. The OP arbiter selects CBs based on a round-robin manner. Upon selection of a CB, the HOL cell is forwarded from the CB to the OP.

Table 2.1 Notations used in the Description of the LBC Switch

Term	Description
N	Number of input/output ports.
n	Number of input/output ports for each IM and OM.
m	Number of CIMs and COMs.
k	Number of IMs and OMs, where $k = \frac{N}{n}$.
$IP(i, s)$	Input port s of $IM(i)$, where $0 \leq i \leq k-1, 0 \leq s \leq n-1$.
$IM(i)$	Input module i.
$OM(j)$	Output module j, where $0 \leq j \leq k-1$.
$CIM(r)$	Central Input Module r, where $0 \leq r \leq m-1$.
$COM(r)$	Central Output Module r.
$VOQ(i, s, j, d)$	VOQ at $IP(i, s)$ that stores cells destined to $OP(j, d)$, where $0 \leq d \leq n-1$.
$L_{IM}(i, r)$	Output link of $IM(i)$ connected to $CIM(r)$.
$L_{CIM}(r, p)$	Output port p of $CIM(r)$, where $0 \leq p \leq k-1$.
$I_C(r, p)$	Input port p of $COM(r)$.
$L_{COM}(r, j)$	Output link of $COM(r)$ connected to $OM(j)$.
$VOMQ(r, p, j)$	VOMQ at output of CIMs that stores cells destined to $OM(j)$.
$VOPQ(r, p, j, d)$	VOPQ at output of CIMs that stores cells destined to $OP(j, d)$.
$CB(r, j, d)$	Crosspoint buffer at $OM(j)$ that stores cells going through $COM(r)$ and destined to $OP(j, d)$.
$OP(j, d)$	Output port d at $OM(j)$.

2.2.1 Module Configuration

The IMs and CIMs in the IM-CIM stage are configured based on a pre-determined sequence of disjoint permutations, applying one permutation every time slot. We call a permutation disjoint from the set of permutations if an input-output pair is interconnected in one and only one of the permutations. This pre-determined sequence of permutations repeats every k time slots. Cells at the inputs of IMs are

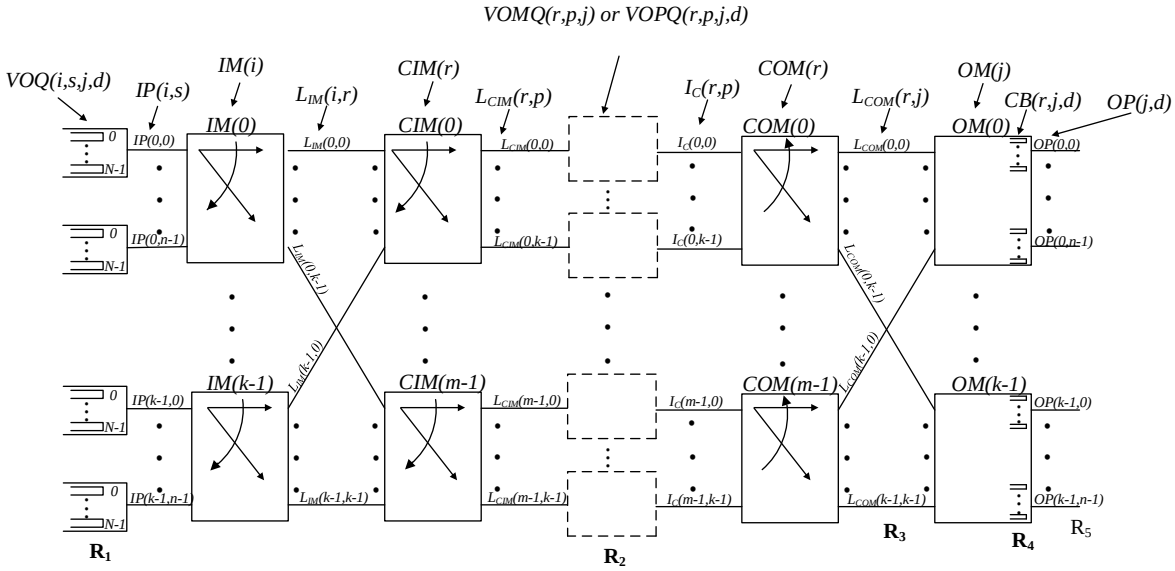
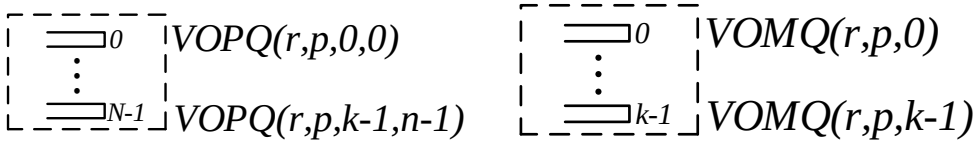


Figure 2.1 Split-central buffered load-balancing Clos-network (LBC) switch.



(a) VOPQs. (b) VOMQs

Figure 2.2 Split-central buffers with: (a) VOPQs and (b) VOMQs.

forwarded to the outputs of the CIMs determined by the configuration of that time slot. A cell is then stored in the VOMQ corresponding to its destination OM.

The COMs follow a configuration similar to that of the CIMs, but in a mirror (i.e., reverse order) sequence. The HoL cell at the VOMQ destined to $OM(j)$ is forwarded to its destination when the input of the COM is connected to the input of the destination $OM(j)$. Else, the HoL cell waits until the required configuration takes place. The forwarded cell is queued at the CB of its destination OP once it arrives in the OM. At the OP, a CB (i.e., HoL cell of that queue) is selected from all non-empty CBs by an output arbitration scheme.

The specific configurations of the bufferless modules, IM, CIM, COM, and OM are as follows.

At time slot t, $IM(i)$ is configured to interconnect input $IP(i,s)$ to $L_{IM}(i,r)$, with:

$$r = (s + t) \mod m \tag{2.1}$$

Similarly, CIM input $L_{IM}(i,r)$ is interconnected to CIM output $L_{CIM}(r,p)$ at time slot t with:

$$p = (i + t) \mod k \tag{2.2}$$

The configuration of COMs is similar to that of IMs, but in a reverse sequence. At time slot t, COM input $I_C(r,p)$ is interconnected to output $L_{COM}(r,j)$ with:

$$j = (p - t) \mod k \tag{2.3}$$

Recall that $a \; mod \; k = a + (mutiples \; of \; k) > 0$ when $a < 0$ (e.g., $-2 \; mod \; 5 = 3$)

Round-robin could also be used to select VOMQs and configure COMs. OM buffers allow forwarding a cell from a VOMQ to the destination output without requiring port matching [48].

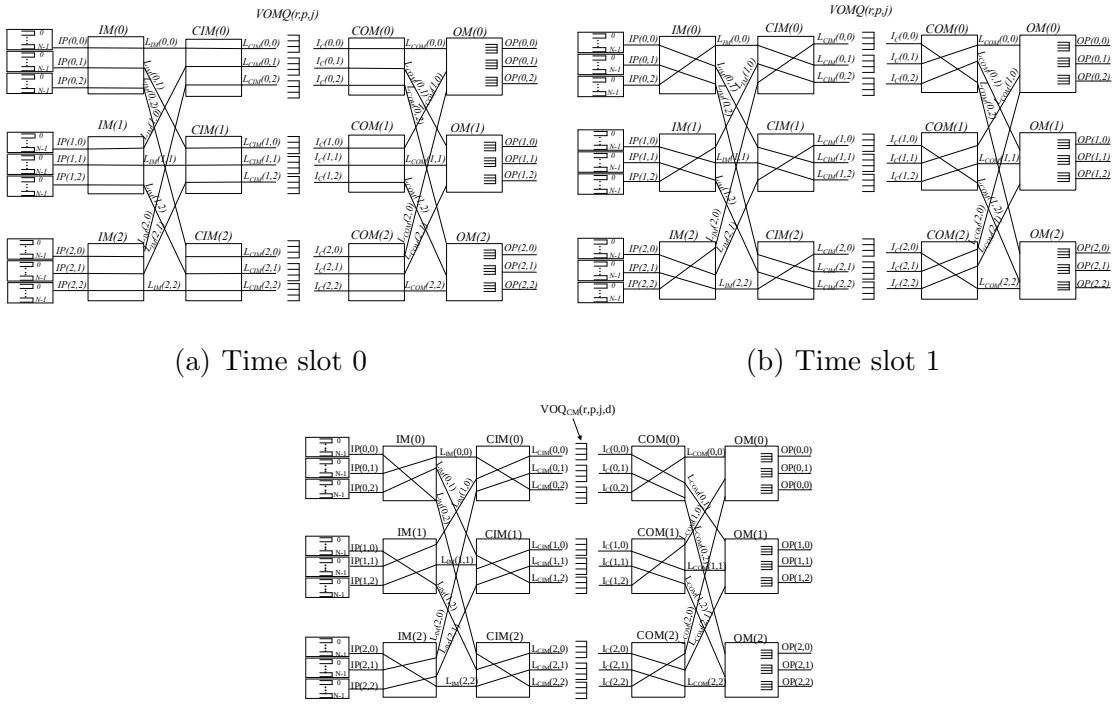
Figure 2.3 shows an example of the configuration of a 9×9 LBC switch. As $k = 3$, the example shows the configuration of three consecutive time slots, after which the configuration pattern repeats. Because similar connections are set for all the IMs and CIMs and a different connection pattern is set for all COMs at each time slot, Table 2.2 describes the configuration on the figure for $IM(0)$, $CIM(0)$, and $COM(0)$ at each time slot. In this example, we use $\rightarrow$ to denote an interconnection.

Table 2.2 Example of Configuration of Modules in a 9×9 LBC Switch

Configuration			
Time slot	$IM(0)$	$CIM(0)$	$COM(0)$
$t = 0$	$IP(0,0) \rightarrow L_{IM}(0,0)$	$L_{IM}(0,0) \rightarrow L_{CIM}(0,0)$	$I_c(0,0) \rightarrow L_{COM}(0,0)$
	$IP(0,1) \rightarrow L_{IM}(0,1)$	$L_{IM}(1,0) \rightarrow L_{CIM}(0,1)$	$I_c(0,1) \rightarrow L_{COM}(0,1)$
	$IP(0,2) \rightarrow L_{IM}(0,2)$	$L_{IM}(2,0) \rightarrow L_{CIM}(0,2)$	$I_c(0,0) \rightarrow L_{COM}(0,2)$
$t = 1$	$IP(0,0) \rightarrow L_{IM}(0,1)$	$L_{IM}(0,0) \rightarrow L_{CIM}(0,1)$	$I_c(0,0) \rightarrow L_{COM}(0,2)$
	$IP(0,1) \rightarrow L_{IM}(0,2)$	$L_{IM}(1,0) \rightarrow L_{CIM}(0,2)$	$I_c(0,1) \rightarrow L_{COM}(0,0)$
	$IP(0,2) \rightarrow L_{IM}(0,0)$	$L_{IM}(2,0) \rightarrow L_{CIM}(0,0)$	$I_c(0,2) \rightarrow L_{COM}(0,1)$
$t = 2$	$IP(0,0) \rightarrow L_{IM}(0,2)$	$L_{IM}(0,0) \rightarrow L_{CIM}(0,2)$	$I_c(0,0) \rightarrow L_{COM}(0,1)$
	$IP(0,1) \rightarrow L_{IM}(0,0)$	$L_{IM}(1,0) \rightarrow L_{CIM}(0,0)$	$I_c(0,1) \rightarrow L_{COM}(0,2)$
	$IP(0,2) \rightarrow L_{IM}(0,1)$	$L_{IM}(2,0) \rightarrow L_{CIM}(0,1)$	$I_c(0,2) \rightarrow L_{COM}(0,0)$

2.2.2 Arbitration at Output Ports

An output port arbiter selects a HoL cell from the crosspoint buffers in a round-robin fashion. Because there is one cell from each flow at these buffers, out-of-sequence forwarding is not a concern at this stage. We discuss this case in Section 2.4. Here, a flow is the set of cells from $IP(i, s)$ destined to $OP(j, d)$. The round-robin schedule ensures fair service for different flows.



(a) Time slot 0

(b) Time slot 1

(c) Time slot 2

Figure 2.3 Configuration example of LBC switch modules.

2.2.3 In-sequence Cell Forwarding Mechanism

The proposed in-sequence forwarding mechanism for the LBC switch is based on holding cells of a flow at the VOQs so that no younger cell is forwarded from VOMQs to OPs before any given cell of the same flow. The policy used for holding cells at an IP is as follows: No cell of flow y at the IP is forwarded to a VOMQ for δk time slots after cell τ of the same flow has been forwarded to a VOMQ, whose occupancy is δ cells at the time of arrival at the VOMQ. For a cell that arrives at an empty VOMQ, $\delta = 0$. The flow control mechanism keeps IPs informed about VOMQ occupancy as discussed in Section 2.2.5.

Figure 2.4 shows an example of this forwarding mechanism for flow A. Cells from flow A are denoted as A_t, where t is the cell arrival time. In this example, cells arrive at time slots 1, 2, 4, and 5, and they are denoted as A_1, A_2, A_4, and A_5, respectively. VOMQ(k) denotes the kth VOMQ to where cells are forwarded. Here,

the "X" mark indicates that the buffer at VOMQ(k) is occupied by cells from other flows. Assuming $k = 3$ and no other cell arrival or departure during this time period, A_1 is the first cell of the flow with arrival time $t = 1$ and is sent to VOMQ(1) at time slot $t = 2$. Because VOMQ(1) has no backlogged cells before A_1, there is no waiting time for A_2. Therefore, A_2 is sent to VOMQ(2) at $t = 3$. A_2 finds three cells already queued, so no cell from this flow is forwarded in $3 * 3 = 9$ time slots, or from time slots $t = 4$ to $t = 12$. After that, A_4 is sent to VOMQ(3) at $t = 13$. This cell finds no other cell, so A_5 is sent to VOMQ(1) at $t = 14$.

2.2.4 Implementation of In-sequence Mechanism

Each IP has an input port counter (IPC) for each VOMQ to which it connects. IPCs keep track of the number of cells at these VOMQs. Each IP also has a hold-down timer for each VOQ. The timer is used by the in-sequence forwarding mechanism. The timer is triggered by the IPC count of the VOMQ where the last cell was forwarded. When a cell is forwarded from a VOQ to VOMQ, and the IPC is updated to σ, this update sets the hold-down timer for that VOQ for $(\sigma-1)k$ time slots, where $\delta = \sigma-1$.

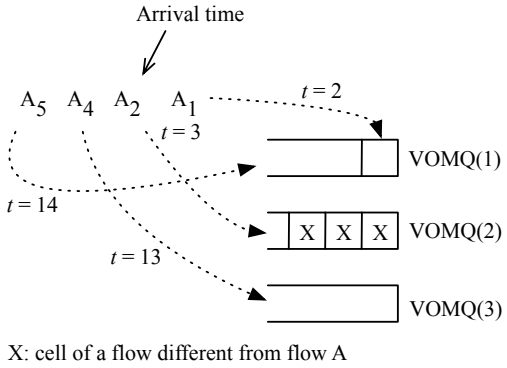


Figure 2.4 Example of the operation of the proposed in-sequence forwarding mechanism.

2.2.5 Flow Control

There is a flow control mechanism between VOMQs and IPs and another between CBs and VOMQs that extends to IPs. There are fixed connections between each VOMQ and its k corresponding IPs and between each CB and its corresponding k I_Cs. Each IP has $mk = N$ occupancy counters, IPCs, one per VOMQ. Each VOMQ updates the corresponding k IPCs about its occupancy. A VOMQ uses two thresholds for flow control; pause (T_{pv}) and resume (T_{rv}), where $T_{pv} > T_{rv}$, in number of cells. When the occupancy of VOMQ, $|VOMQ|$, is larger than T_{pv}, the VOMQ signals the corresponding IPs to pause sending cells to it. When the $|VOMQ| < T_{rv}$, the VOMQ signals the corresponding IPs to resume sending cells to it. Here, T_{pv} is such that $C_{VOMQ} - T_{pv} \geq D_v$, where C_{VOMQ} is the size of the VOMQ and D_v is the flow-control information delay.

Similar to VOMQs, CBs use two thresholds; pause (T_{pc}) and resume (T_{rc}), where $T_{pc} > T_{rc}$, and T_{pc} is such that $C_{CB} - T_{pc} \geq D_c$, where C_{CB} is the capacity of a CB and D_c is the flow-control information delay between a CB and corresponding IPs. These CB thresholds work in a similar way as for VOMQs. Different from IPs, VOMQs have a binary flag to pause/resume forwarding of cells to CBs. When the occupancy of a CB, $|CB|$, becomes larger than T_{pc}, the CB informs the corresponding VOMQs, and in turn VOMQs inform corresponding IPs to pause forwarding cells to the VOMQ for the congested OP. With IPs paused for traffic to a CB, traffic already at VOMQs can still be forwarded to CBs as long as $|CB|$ is such that $T_{pc} < |CB| < C_{CB}$. When $|CB| < T_{rc}$, the CB signals the corresponding VOMQs to resume forwarding, and in turn, VOMQs signal source IPs to resume forwarding cells for that destination OP.

2.2.6 Avoiding HoL Blocking in LBC with VOMQs

Concerns of HoL blocking, owning to the aggregation of traffic going to different OPs at the same OM at a VOMQ, may arise. However, one must note that this HoL

blocking may occur if and only if a CB gets congested. Here, we argue that the efficient load-balancing mechanism and the use of one CB for each COM at an OP avoids congestion of CBs even in the presence of heavy (but admissible) traffic. We also show that CB occupancy does not build up. Let us consider the input traffic matrix, $\mathbf{R_1}$, with input load, $\lambda_{i,s,j,d}$, which gets load-balanced to CIMs at rate of $\frac{\lambda_{i,s,j,d}}{m}$. The aggregate traffic arrival rate at an L_{CIM} from all IMs, R_{LCIM}, is:

$$R_{LCIM} = \frac{1}{m} \sum_{i=0}^{k} \lambda_{i,s,j,d} \tag{2.4}$$

Therefore, the traffic arrival rate to a CB from COMs, R_{CB}, is:

$$R_{CB} = \frac{1}{mk} \sum^{k} \sum_{i=0}^{k} \lambda_{i,s,j,d} \tag{2.5}$$

To test the growth of CBs, we consider three stressing traffic scenarios: a) All IPs in the switch have traffic only for OPs in an OM; b) all IPs in an IM forward traffic to all OPs in an OM; and c) a single flow, with a large rate, going from an IP to a single OP.

Then, for a) the largest arrival rate at IPs while being admissible is:

$$\lambda_{i,s,j,d} = \frac{1}{N} \tag{2.6}$$

Substituting (2.6) into (2.5) and $m = n = k$ yields:

$$R_{CB} = \frac{1}{k^2} \sum^{k} \sum_{i=0}^{k} \frac{1}{N} = \frac{1}{N} = \frac{1}{k^2} \tag{2.7}$$

Because round-robin is used as selection policy at an OP, the service rate, S_{CB}, of a CB would be:

$$\frac{1}{k} \leq S_{CB} \leq 1$$

Yet, while considering the worst case scenario, or:

$$S_{CB} = \frac{1}{k} \tag{2.8}$$

Therefore, CB occupancy does not grow because $S_{CB} > R_{CB}$.

For b), the arrival rate at IPs for admissibility is:

$$\lambda_{i,s,j,d} = \frac{1}{k} \tag{2.9}$$

Substituting (2.9) into (2.5) yields:

$$R_{CB} = \frac{1}{m}\frac{1}{k}\sum^{k}\sum_{i=0}^{k}\frac{1}{k} = \frac{1}{k} \tag{2.10}$$

The service rate would be the same as in (2.8). Therefore, the CB would not become congested as $R_{CB} = S_{CB}$.

For c), the arrival rate at the IP:

$$\lambda_{i,s,j,d} = 1 \tag{2.11}$$

The traffic arrival rate to an L_{CIM} is:

$$R_{LCIM} = \frac{1}{m}\lambda_{i,s,j,d} = \frac{1}{m} \tag{2.12}$$

The traffic arrival rate to a CB from COMs is:

$$R_{CB} = \frac{1}{m}\frac{1}{k}\sum^{k} = \frac{1}{m} = \frac{1}{k} \tag{2.13}$$

Therefore, the CB would not become congested because $R_{CB} \leq S_{CB}$ for admissible traffic.

2.3 Throughput Analysis

In this section, we analyze the performance of the proposed LBC switch. Let us denote the traffic coming to the IM-CIM stage, the COM stage, the OMs, OPs, and

the traffic leaving LBC as $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, $\mathbf{R_4}$, and R_5, respectively. Figure 2.1 shows these analysis points. Here, $\mathbf{R_1}$, $\mathbf{R_2}$, and $\mathbf{R_3}$ are $N \times N$ matrices, $\mathbf{R_4}$ comprises N $m \times 1$ column vectors, and R_5 comprises N scalars.

The traffic from input ports to the IM-CIM stage, $\mathbf{R_1}$, is defined as:

$$\mathbf{R_1} = [\lambda_{u,v}] \tag{2.14}$$

Here, $\lambda_{u,v}$ is the arrival rate of traffic from input u to output v, where

$$u = ik + s \tag{2.15}$$

$$v = jm + d \tag{2.16}$$

and $0 \leq u, v \leq N - 1$.

In the following analysis, we consider admissible traffic, which is defined as:

$$\sum_{u=0}^{N-1} \lambda_{u,v} \leq 1, \ \sum_{v=0}^{N-1} \lambda_{u,v} \leq 1 \tag{2.17}$$

under i.i.d. traffic conditions.

The IM-CIM stage of the LBC switch balances the traffic load coming from the input ports to the VOMQs. Specifically, the permutations used to configure the IMs and CIMs interconnect the traffic from an input to k different CIMs, and then to the VOMQs connected to these CIMs.

$\mathbf{R_2}$ is the traffic directed towards COMs and it is derived from $\mathbf{R_1}$ and the permutations of IMs and CIMs. The configuration of the combined IM-CIM stage at time slot t that connects $IP(i, s)$ to $L_{CIM}(r, p)$ are represented as an $N \times N$ permutation matrix, $\mathbf{\Pi}(t) = [\pi_{u,v}]$, where r and p are determined from (2.1) and (2.2) and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = rk + p \\ 0 & \text{elsewhere.} \end{cases}$$

The configuration of the compound IM-CIM stage can be represented as a compound permutation matrix, $\mathbf{P_1}$, which is the sum of the IM-CIM permutations over k time slots as follows,

$$\mathbf{P_1} = \sum^{k} \mathbf{\Pi}(t) \tag{2.18}$$

Because the configuration is repeated every k time slots, the traffic load from the same input going to each VOMQ is $\frac{1}{k}$ of the traffic load of $\mathbf{R_1}$. Therefore, a row of $\mathbf{R_2}$ is the sum of the row elements of $\mathbf{R_1}$ at the non zero positions of $\mathbf{P_1}$, normalized by k. This is:

$$\mathbf{R_2} = \frac{1}{k}((\mathbf{R_1} * \mathbb{1}) \circ \mathbf{P_1}) \tag{2.19}$$

where $\mathbb{1}$ denotes an $N \times N$ unit matrix and $\circ$ denotes element/position wise multiplication.

There are k non-zero elements in each row of $\mathbf{R_2}$. Here, $\mathbf{R_2}$ is the aggregate traffic in all the VOMQs destined to all OPs. This matrix can be further decomposed into k $N \times N$ submatrices, $\mathbf{R_2}(j)$, each of which is the aggregate traffic at VOMQs designated for $OM(j)$.

$$\mathbf{R_2} = \sum_{j=0}^{j=k-1} \mathbf{R_2}(j) \tag{2.20}$$

where j is obtained from (2.16) for all d. The configuration of the COM stage at time slot t that connects $I_c(r,p)$ to $L_{COM(r,j)}$ can be represented as an $N \times N$ permutation

matrix, $\mathbf{\Phi}(t) = [\phi_{u,v}]$, and the matrix element;

$$\phi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = jk + r \\ 0 & \text{elsewhere.} \end{cases} \tag{2.21}$$

Similarly, the switching at the COM stage is represented by a compound permutation matrix $\mathbf{P_2}$, which is the sum of k permutations of the COM stage over k time slots. Here

$$\mathbf{P_2} = \sum^{k} \mathbf{\Phi}(t) \tag{2.22}$$

The output traffic of COMs going to different OMs is described by matrix $\mathbf{R_3}(j)$, which is defined as

$$\mathbf{R_3}(j) = \mathbf{R_2}(j) \circ \mathbf{P_2} \tag{2.23}$$

where j is obtained from (2.16) for all d. The traffic destined to $OP(j,d)$ at $OM(j)$, $\mathbf{R_3}(j,d)$, is obtained by extracting the traffic elements from $\mathbf{R_3}(j)$, or:

$$\mathbf{R_3}(j) = \sum_{d=0}^{d=k-1} \mathbf{R_3}(j,d) \tag{2.24}$$

where d is obtained from (2.16) for the different j.

$\mathbf{D_s}$ is an $m \times N$ matrix, built by concatenating N $k \times 1$ vector of all ones, $\vec{1}$, as:

$$\mathbf{D_s} = [\vec{1}, \cdots, \vec{1}] \tag{2.25}$$

$\vec{A}$ is a $1 \times k$ row vector, built by setting the first element to 1 and every other element to 0, or:

$$\vec{A} = [1 \cdots 0] \tag{2.26}$$

$\vec{A_s}$ is an $N \times 1$ column vector, built by concatenating k $\vec{A}$ and taking the transpose, or:

$$\vec{A_s} = [\vec{A_{s_1}}, \cdots, \vec{A_{s_k}}]^T \tag{2.27}$$

where $\vec{A_{s_1}} = \vec{A_{s_k}} = \vec{A}$, such that

$$\vec{A_s} = [\vec{A}, \cdots, \vec{A}]^T \tag{2.28}$$

The traffic queued at the CB of an OP, $\mathbf{R_4}(v)$, is the multiplication of $\mathbf{D_s}$, $\mathbf{R_3}(j,d)$, and $\vec{A_s}$, or:

$$\mathbf{R_4}(v) = \mathbf{D_s} * \mathbf{R_3}(j,d) * \vec{A_s} \tag{2.29}$$

The traffic leaving an OP, $R_5(v)$, is:

$$R_5(v) = (\vec{1})^T * \mathbf{R_4}(v) \tag{2.30}$$

Therefore, $\mathbf{R_5}(v)$ is the sum of the traffic leaving $OP(v)$.

Equations (2.19), (2.29), and (2.30) show that the admissibility conditions in (2.17) are satisfied by the traffic at the VOMQ, CBs, and OP. Because $\mathbf{R_2}$, $\mathbf{R_4}(v)$, and $R_5(v)$ meet the admissibility conditions in (2.17), this implies that the sum of the traffic load at each $VOMQ$, CB, and OP does not exceed their respective capacities. From (2.29), we can deduce that $\mathbf{R_4}$ is equal to the input traffic $\mathbf{R_1}$, or:

$$\mathbf{R_4}(v) = \mathbf{R_1}(v) \; \forall \; v \tag{2.31}$$

From the admissibility of $\mathbf{R_2}$, $\mathbf{R_4}(v)$, $R_5(v)$ and (2.31), we can infer that the input traffic is successfully forwarded to the output ports.

As discussed in Section 2.2.2, the output arbiter selects a flow in a round-robin fashion and if no cell of a flow is selected, the OP arbiter moves to the next flow. This implies the queues are work conserving which ensures fairness and that cells forwarded to OPs are successfully forwarded out of OPs. Hence, from (2.30), we can infer that $R_5(v)$ is equal to $\mathbf{R_4}(v)$, or:

$$R_5(v) = (\vec{1})^T * \mathbf{R_4}(v) \; \forall \; v \tag{2.32}$$

From (2.31) and (2.32), we can conclude that LBC successfully forwards all traffic at IPs out of OPs.

The following example shows the different traffic matrices for a 4×4 ($k = 2$) LBC switch. Let the input traffic matrix be

$$\mathbf{R_1} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix}$$

First, $\mathbf{R_1}$ is decomposed into $\mathbf{R_2}$ at the IM-CIM stage. From (2.18), the compound permutation matrix for the IM-CIM stage for this switch is:

$$\mathbf{P_1} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Using (2.19), we get:

$$\mathbf{R_2} = \frac{1}{2} \begin{bmatrix} \sum_{i=0}^{3} \lambda_{0,i} & 0 & 0 & \sum_{i=0}^{3} \lambda_{0,i} \\ 0 & \sum_{i=0}^{3} \lambda_{1,i} & \sum_{i=0}^{3} \lambda_{1,i} & 0 \\ 0 & \sum_{i=0}^{3} \lambda_{2,i} & \sum_{i=0}^{3} \lambda_{2,i} & 0 \\ \sum_{i=0}^{3} \lambda_{3,i} & 0 & 0 & \sum_{i=0}^{3} \lambda_{3,i} \end{bmatrix}$$

From (2.20), the traffic matrix at VOMQs destined for the different OMs are:

$$\mathbf{R_2}(0) = \frac{1}{2} \begin{bmatrix} \lambda_{0,0} + \lambda_{0,1} & 0 & 0 & \lambda_{0,0} + \lambda_{0,1} \\ 0 & \lambda_{1,0} + \lambda_{1,1} & \lambda_{1,0} + \lambda_{1,1} & 0 \\ 0 & \lambda_{2,0} + \lambda_{2,1} & \lambda_{2,0} + \lambda_{2,1} & 0 \\ \lambda_{3,0} + \lambda_{3,1} & 0 & 0 & \lambda_{3,0} + \lambda_{3,1} \end{bmatrix}$$

$$\mathbf{R_2}(1) = \frac{1}{2}\begin{bmatrix} \lambda_{0,2}+\lambda_{0,3} & 0 & 0 & \lambda_{0,2}+\lambda_{0,3} \\ 0 & \lambda_{1,2}+\lambda_{1,3} & \lambda_{1,2}+\lambda_{1,3} & 0 \\ 0 & \lambda_{2,2}+\lambda_{2,3} & \lambda_{2,2}+\lambda_{2,3} & 0 \\ \lambda_{3,2}+\lambda_{3,3} & 0 & 0 & \lambda_{3,2}+\lambda_{3,3} \end{bmatrix}$$

The rows of $\mathbf{R_2}(v)$ represent the traffic from IPs, and the columns represent $VOMQ(r,p,j)$ at $I_C(r,p)$. From (2.22), the compound permutation matrix for the COM stage for this switch is:

$$\mathbf{P_2} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

From (2.23) and (2.24), the traffic forwarded to an OP is:

$$\mathbf{R_3}(0,0) = \frac{1}{2}\begin{bmatrix} \lambda_{0,0} & 0 & 0 & \lambda_{0,0} \\ 0 & \lambda_{1,0} & \lambda_{1,0} & 0 \\ 0 & \lambda_{2,0} & \lambda_{2,0} & 0 \\ \lambda_{3,0} & 0 & 0 & \lambda_{3,0} \end{bmatrix}$$

$$\mathbf{R_3}(0,1) = \frac{1}{2}\begin{bmatrix} \lambda_{0,1} & 0 & 0 & \lambda_{0,1} \\ 0 & \lambda_{1,1} & \lambda_{1,1} & 0 \\ 0 & \lambda_{2,1} & \lambda_{2,1} & 0 \\ \lambda_{3,1} & 0 & 0 & \lambda_{3,1} \end{bmatrix}$$

$$\mathbf{R_3}(1,0) = \frac{1}{2}\begin{bmatrix} \lambda_{0,2} & 0 & 0 & \lambda_{0,2} \\ 0 & \lambda_{1,2} & \lambda_{1,2} & 0 \\ 0 & \lambda_{2,2} & \lambda_{2,2} & 0 \\ \lambda_{3,2} & 0 & 0 & \lambda_{3,2} \end{bmatrix}$$

$$\mathbf{R_3}(1,1) = \frac{1}{2}\begin{bmatrix} \lambda_{0,3} & 0 & 0 & \lambda_{0,3} \\ 0 & \lambda_{1,3} & \lambda_{1,3} & 0 \\ 0 & \lambda_{2,3} & \lambda_{2,3} & 0 \\ \lambda_{3,3} & 0 & 0 & \lambda_{3,3} \end{bmatrix}$$

The rows of $\mathbf{R_3}(j,d)$ represent the traffic from $VOMQ(r,p,j)$ at $I_C(r,p)$ and the columns represent $L_{COM}(r,j)$. $\mathbf{D_S}$ and $\vec{A}_s$ are obtained from (2.25) and (2.28), respectively, as:

$$\mathbf{D_s} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\vec{A}_s = [1\ 0\ 1\ 0]^T$$

The traffic forwarded from CBs to the corresponding OP is obtained from (2.29):

$$\mathbf{R_4}(0) = \frac{1}{2}\begin{bmatrix} \sum_{i=0}^{3} \lambda_{i,0} \\ \sum_{i=0}^{3} \lambda_{i,0} \end{bmatrix}$$

$$\mathbf{R_4}(1) = \frac{1}{2}\begin{bmatrix} \sum_{i=0}^{3} \lambda_{i,1} \\ \sum_{i=0}^{3} \lambda_{i,1} \end{bmatrix}$$

$$\mathbf{R_4}(2) = \frac{1}{2}\begin{bmatrix} \sum_{i=0}^{3} \lambda_{i,2} \\ \sum_{i=0}^{3} \lambda_{i,2} \end{bmatrix}$$

$$\mathbf{R_4}(3) = \frac{1}{2}\begin{bmatrix} \sum_{i=0}^{3} \lambda_{i,3} \\ \sum_{i=0}^{3} \lambda_{i,3} \end{bmatrix}$$

The rows of $\mathbf{R_4}(v)$ represent the traffic from $COM(r)$. Using (2.30), we obtain the sum of the traffic leaving the OP, or:

$$R_5(0) = \sum_{i=0}^{3} \lambda_{i0}$$

$$R_5(1) = \sum_{i=0}^{3} \lambda_{i1}$$

$$R_5(2) = \sum_{i=0}^{3} \lambda_{i2}$$

$$R_5(3) = \sum_{i=0}^{3} \lambda_{i3}$$

We use the traffic analysis of the previous section to demonstrate that the LBC switch achieves 100% throughput under admissible traffic. This demonstration is provided in Appendix 2.3.1.

2.3.1 100% Throughput

In this section we prove that LBC achieves 100% throughput by using the analysis presented on Section 2.3 and the concept of queue stability. A switch is defined as stable for a traffic pattern if the queue length is bounded and a switch achieves 100% throughput if it is stable for admissible i.i.d. traffic [57, 58]. With this, we set the following theorem:

Theorem 1. *LBC achieves 100% throughput under admissible i.i.d traffic.*

Proof: Here, we consider the queue to be weakly stable if the drift of the queue occupancy from the initial state is a finite integer $\epsilon \quad \forall\ t$ as $\lim_{t\to\infty}$. Using the definition above, we show that the queue length of VOQs, VOMQs, and CBs are

weakly stable under i.i.d. traffic, and hence, achieves 100% throughput under that traffic pattern.

Let us represent the queue occupancy of VOQs at time slot t, $\mathbf{N_1}(t)$ as:

$$\mathbf{N_1}(t) = \mathbf{N_1}(t-1) + \mathbf{A_1}(t) - \mathbf{D_1}(t) \tag{2.33}$$

where $\mathbf{A_1}(t)$ is the packet arrival matrix at time slot t to VOQs and $\mathbf{D_1}(t)$ is the service rate matrix of VOQs at time slot t. Solving (2.33) with an initial condition $\mathbf{N_1}(0)$, recursively yields:

$$\mathbf{N_1}(t) = \mathbf{N_1}(0) + \sum_{\gamma=0}^{t} \mathbf{A_1}(\gamma) - \sum_{\gamma=0}^{t} D_1(\gamma) \tag{2.34}$$

Let us consider $s_{1_{u,v}}(t)$ as the service rate received by the VOQ at $IP(u)$ for $OP(v)$ at time slot t or:

$$\begin{cases} \dfrac{1}{N} \leq s_{1_{u,v}}(t) \leq 1 & \text{for } \delta = 0 \\ \dfrac{1}{\delta N k} \leq s_{1_{u,v}}(t) \leq \dfrac{1}{\delta k} & \text{for } \sigma > 1 \end{cases} \tag{2.35}$$

Another way to express $D_1(t)$ is:

$$\mathbf{D_1}(t) = [s_{1_{u,v}}(t)] \tag{2.36}$$

and recalling $\mathbf{R_1}$ as the aggregate traffic arrival to VOQs or:

$$\mathbf{R_1} = \sum_{\gamma=0}^{t} \mathbf{A_1}(\gamma) \tag{2.37}$$

Let us consider the worse case scenario in (2.35). Substituting (2.35) into (2.36), and (2.36) and (2.37) into (2.34), yields:

$$\mathbf{N_1}(t) = \begin{cases} \mathbf{N_1}(0) + \mathbf{R_1} - \dfrac{t}{N} * \mathbb{1} & \text{for } \delta = 0 \\ \mathbf{N_1}(0) + \mathbf{R_1} - \dfrac{t}{\delta N k} * \mathbb{1} & \text{for } \delta > 1 \end{cases} \tag{2.38}$$

From (2.38), we obtain:

$$\begin{cases} \lim\limits_{t\to\infty} \dfrac{\mathbf{R_1}}{t} - \dfrac{1}{N} * \mathbb{1} \leq \epsilon < \infty & \text{for } \delta = 0 \\ \lim\limits_{t\to\infty} \dfrac{\mathbf{R_1}}{t} - \dfrac{1}{\delta N k} * \mathbb{1} \leq \epsilon < \infty & \text{for } \delta > 1 \end{cases} \tag{2.39}$$

where ϵ is a finite real number that determines the drift from the initial occupancy. From the admissibility condition of $\mathbf{R_1}$, it is easy to see that for any value of t, (2.39) is finite. Hence, from the admissibility of $\mathbf{R_1}$, (2.38) and (2.39), we conclude that occupancy of VOQ is weakly stable.

■

Now we prove VOMQs stability. As before, the queue occupancy matrix of VOMQs at time slot t can be represented as:

$$\mathbf{N_2}(t) = \mathbf{N_2}(t-1) + \mathbf{A_2}(t) - \mathbf{D_2}(t) \tag{2.40}$$

where $\mathbf{A_2}(t)$ is the arrival matrix at time slot t to VOMQs and $\mathbf{D_2}(t)$ is the service rate matrix of VOMQs at time slot t. Solving (2.40) recursively with consideration of an initial condition for $\mathbf{N_2}(t)$, yields:

$$\mathbf{N_2}(t) = \mathbf{N_2}(0) + \sum_{\gamma=0}^{t} \mathbf{A_2}(\gamma) - \sum_{\gamma=0}^{t} \mathbf{D_2}(\gamma) \tag{2.41}$$

Because a VOMQ is serviced at least once every k time slots, the service rate of the VOMQ at $I_C(r,p)$ for $OP(v)$ at time slot t, $d_{2_{\mu,v}}(t)$ is:

$$d_{2_{\mu,v}}(t) = \frac{1}{k} \; \forall \; \mu \; and \; v$$

Then, the service matrix of VOMQs is:

$$\mathbf{D_2}(t) = [d_{2_{\mu,v}}(t)] \tag{2.42}$$

and representing $\mathbf{R_2}$ as the aggregate traffic arrival to VOMQs or:

$$\mathbf{R_2} = \sum_{\gamma=0}^{t} \mathbf{A_2}(\gamma) \tag{2.43}$$

Substituting (2.42) and (2.43) into (2.41) gives:

$$\mathbf{N_2}(t) = \mathbf{N_2}(0) + \mathbf{R_2} - \frac{1}{k}\mathbf{P_1} \tag{2.44}$$

$$\mathbf{R_2} - \frac{1}{k}\mathbf{P_1} \leq \epsilon < \infty \tag{2.45}$$

Recalling that $\mathbf{R_2}$ is admissible, per the discussion in Section III.A, and by substituting $\mathbf{P_1}$ and $\mathbf{R_2}$ into (2.45), it is easy to see that ϵ is finite. Hence, from (2.44) and (2.45), we conclude that the occupancy of VOMQ is weakly stable.

■

Now we prove the stability of CBs. The queue occupancy matrix of CBs at time slot t can be represented as:

$$\mathbf{N_3}(t) = \mathbf{N_3}(t-1) + \mathbf{A_3}(t) - \mathbf{D_3}(t) \tag{2.46}$$

where $\mathbf{A_3}(t)$ is the packet arrival matrix at time slot t CBs, and $\mathbf{D_3}(t)$ is the service rate matrix of CBs at time slot t. Solving (2.46) recursively as before yields:

$$\mathbf{N_3}(t) = \mathbf{N_3}(0) + \sum_{\gamma=0}^{t} \mathbf{A_3}(\gamma) - \sum_{\gamma=0}^{t} \mathbf{D_3}(\gamma) \tag{2.47}$$

Because a CB is serviced at least once every k time slots. Hence, the service rate of the CB at $OP(v)$ at time slot t, $d_{3_v}(t)$ is:

$$\frac{1}{k} \leq d_{3_v}(t) \leq 1$$

and service matrix of CBs is:

$$\mathbf{D_3}(t) = [d_{3_v}(t)] \tag{2.48}$$

Similarly, the aggregate traffic arrival to the CB or:

$$\mathbf{R_4} = \sum_{\gamma=0}^{t} A_3(\gamma) \tag{2.49}$$

Let us assume $d_{3_v}(t) = \frac{1}{k} \; \forall \; v$ in (2.48), which is the worst case scenario at which a CB gets served once every k time slots. Substituting (2.48) and (2.49) into (2.47) gives:

$$\mathbf{N_3}(t) = \mathbf{N_3}(0) + \mathbf{R_4} - \frac{1}{k} * \vec{1} \tag{2.50}$$

where

$$\mathbf{R_4} - \frac{1}{k} * \vec{1} \leq \epsilon < \infty \tag{2.51}$$

With R4 being admissible, as discussed in Section III.A, and by substituting $\mathbf{R_4}$ into (2.51), it is easy to see that ϵ is finite. Hence, from (2.50) and (2.51), we conclude that the occupancy of CB is also weakly stable.

We have shown in Section III.A that $\mathbf{R_2}$, $\mathbf{R_4}$, R_5 are admissible. We have also shown in Section III.A that the traffic forwarded to IPs, $\mathbf{R_1}$, successfully arrives at the CBs, $\mathbf{R_4}$, and is successfully forwarded out the OP, R_5. In this section, we have shown that the queue occupancy matrix at time slot t is finite because the drift from the initial queue occupancy matrix is bounded.

Therefore, the LBC switch is stable and can achieve 100% throughput under admissible i.i.d traffic.

■

This completes the proof of Theorem 1.

■

2.4 Analysis of In-Sequence Service

In this section, we demonstrate that the LBC switch forwards cells in sequence through the proposed in-sequence forwarding mechanism.

Table 2.3 lists the definition of terms used in the discussion of the properties of the proposed LBC switch. Here, $c_{y,\tau}(i,s,j,d)$ denotes the τth cell of traffic flow y, which comprises cells going from $IP(i,s)$ to $OP(j,d)$ with arrival time t_x. In addition, $t_{a_{y,\tau}}$ denotes the arrival time of $c_{y,\tau}$, and $q_{1_{y,\tau}}$, $q_{2_{y,\tau}}$, and $q_{3_{y,\tau}}$ denote the queuing delays experienced by $c_{y,\tau}$ at $VOQ(i,s,j,d)$, $VOMQ(r,p,j)$, and $CB(r,j,d)$, respectively. The departure times of $c_{y,\tau}$ from these queues are denoted as $d_{1_{y,\tau}}$, $d_{2_{y,\tau}}$, and $d_{3_{y,\tau}}$, respectively. In this chapter, we consider admissible traffic as defined in (2.17).

Here, we claim that the LBC switch forward cells in sequence to the output ports, through the following theorem.

Theorem 2. *For any two cells $c_{y,\tau}(i,s,j,d)$ and $c_{y,\tau'}(i,s,j,d)$, where $\tau < \tau'$, $c_{y,\tau}(i,s,j,d)$ departs the destination output port before $c_{y,\tau'}(i,s,j,d)$.*

Table 2.3 Notations for In-sequence Analysis

$c_{y,\tau}$	The τth cell of flow y from $IP(i,s)$ to $OP(j,d)$.
$t_{a_{y,\tau}}$	Arrival time of $c_{y,\tau}$ in $VOQ(i,s,j,d)$ at $IP(i,s)$.
$q_{1_{y,\tau}}$	Queuing delay of $c_{y,\tau}$ at $VOQ(i,s,j,d)$.
$d_{1_{y,\tau}}$	Departure time of $c_{y,\tau}$ from $VOQ(i,s,j,d)$ at $IP(i,s)$.
$q_{2_{y,\tau}}$	Queuing delay of $c_{y,\tau}$ at $VOMQ(r,p,j)$.
$d_{2_{y,\tau}}$	Departure time of $c_{y,\tau}$ from $VOMQ(r,p,j)$ at $L_{COM}(r,j)$.
$q_{3_{y,\tau}}$	Queuing delay of $c_{y,\tau}$ at $CB(r,j,d)$ of $OP(j,d)$.
$d_{3_{y,\tau}}$	Departure time of $c_{y,\tau}$ from $CB(r,j,d)$.

This theorem is sectioned into the following lemmas.

Lemma 1. *For a single flow traversing the LBC switch, any cell of the flow experiences the same delay. This is, let t_d be the delay experienced by a cell. Then, for any cell traversing the LBC switch, $t_{d_{y,\tau}} = \gamma$, where γ is a positive constant.*

A constant delay for each cell implies that cells depart the switch in the same order they arrived under the conditions of this lemma.

We analyze first the scenario of a single flow, i.e., y, traversing the switch, whose cells arrive back to back, one each time slot. For simplicity but without losing generality, let us also consider empty queues as an initial condition.

Proof:

For any $c_{y,\tau}$, the total delay time is defined as:

$$t_{d_{y,\tau}} = q_{1_{y,\tau}} + q_{2_{y,\tau}} + q_{3_{y,\tau}} \tag{2.52}$$

in number of time slots. Here we consider fixed arbitration time at each queue and this delay is included in the queuing delay. We are then interested in finding $q_{1_{y,\tau}}$, $q_{2_{y,\tau}}$, and $q_{3_{y,\tau}}$.

For $q_{1_{y,\tau}}$, under a single-flow scenario, let us consider any two cells of $c_{y,\tau}$ with arrival times k time slots apart, $c_{y,\tau-2k}$ and $c_{y,\tau-k}$, they are forwarded to the same VOMQ. Then, $c_{y,\tau}$ is held at the VOQ (owing to the mechanism to keep cells in sequence at the VOQ) if $c_{y,\tau-k}$ finds one or more cells in the VOMQ, $q_{1_{y,\tau}}$ increases. In this case, the empty queue initial condition makes the waiting factor $\delta = 0$.

On the other hand, an OM is connected to a VOMQ every k time slots as per the configuration scheme of COM. Therefore,

$$q_{2_{y,\tau}} \leq k - 1 \tag{2.53}$$

This queuing delay is smaller than the arrival gap between these two cells as:

$$a_{y,\tau-2k} - a_{y,\tau-k} = k \text{ time slots}$$

Therefore, $c_{y,\tau}$ is not backlogged further in VOMQ and there is no impact on the time the cell is held in a VOQ, such that:

$$q_{1_{y,\tau}} = 0 \ \ \forall \ y, \tau$$

For $q_{2_{y,\tau}}$, let us now assume that $c_{y,\tau-k}$ arrives at a time that it has to wait γ time slots, where $1 \leq \gamma \leq k$, to be forwarded to the destination OM, or

$$q_{2_{y,\tau-k}} = \gamma$$

Then when $c_{y,\tau}$ arrives, k time slots later, it finds exactly the same configuration in the COM as found by $c_{y,\tau-k}$. Because cells arrive consecutively,

$$q_{2_{y,\tau}} = \gamma \ \ \forall \ \tau$$

For $q_{3_{y,\tau}}$, because there is a single flow traversing the switch and the configuration scheme followed by COM, one cell arrives in the CB each time slot and one cell departs OP at the same time slot. Therefore, no cell is backlogged in this case and

$$q_{3_{y,\tau}} = 0$$

From (2.52):

$$t_{d_{y,\tau}} = \gamma \ \ \forall \ \tau$$

for empty queues as initial condition.

It is then easy to see that for any queued cells, $q_{1_{y,\tau}}$ would be increased by δk time slots, and $q_{2_{y,\tau}}$ as well as $q_{3_{y,\tau}}$ remain unchanged.

Therefore, all cells of the flow experience the same delay and are forwarded in sequence.

■

Lemma 2. *For any number of flows traversing the LBC switch, cells from the same flow arrive at the OM in sequence.*

Proof: Here, we consider the following traffic scenario: There are k flows coming from different IPs, each from a different IM. In each of the flows, cells arrive back to back and are destined to the same OP. Furthermore, the flows have one time slot difference in their arrival times such that the cells with the same sequence number of each different flow are stored in the same VOMQs. Here, each flow consists of k cells. Table 2.4 shows an example of the arrival pattern of this traffic scenario for three flows. The table shows the arrival of k cells from k flows at different IPs and IMs that arrive at one time slot apart to enable these flows to be forwarded to the same VOMQ, otherwise the flows would be forwarded to different VOMQs.

Table 2.4 Example of Back-to-back Arrivals of One Burst of k Flows

Cell arrival time				
t_x	t_{x+1}	t_{x+2}	t_{x+3}	t_{x+4}
$c_{1,1}$	$c_{1,2}$	$c_{1,3}$		
	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	
		$c_{3,1}$	$c_{3,2}$	$c_{3,3}$

Table 2.5 shows that cells $c_{1,1}$, $c_{1,2}$, $c_{1,3}$, $c_{2,1}$, and $c_{3,1}$ were successfully forwarded to the VOMQ without any blocking. While the in-sequence mechanism holds back the cells $c_{2,2}$, $c_{2,3}$, $c_{3,2}$ and $c_{3,3}$ to prevent out-of-sequence, because cells $c_{2,1}$ and $c_{3,1}$ were forwarded to a non-empty VOMQ.

The configuration pattern used in the IMs and CIMs, and the in-sequence mechanism determine the order in which cells arrive to the VOMQs. Table 2.5 shows this order in our example.

Table 2.5 Time Slots in which Cells Arrive to VOMQs of a Single k-cell Burst

Time Slots cells arrive at the VOMQs											
t_x	t_{x+1}	t_{x+2}	t_{x+3}	t_{x+4}	t_{x+5}	t_{x+6}	t_{x+7}	t_{x+8}	t_{x+9}	t_{x+10}	t_{x+11}
	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$								
		$c_{2,1}$				$c_{2,2}$	$c_{2,3}$				
			$c_{3,1}$							$c_{3,2}$	$c_{3,3}$

In such arrival pattern, the departures from VOMQs follow the deterministic configuration of the COMs. Table 2.6 shows the corresponding departures of the cells from VOMQs of these three flows.

Table 2.6 Time Slots when Cells Depart VOMQs in Example of the In-sequence Forwarding Mechanism

Cell departure time slots from VOMQs												
t_x	t_{x+1}	t_{x+2}	t_{x+3}	t_{x+4}	t_{x+5}	t_{x+6}	t_{x+7}	t_{x+8}	t_{x+9}	t_{x+10}	t_{x+11}	t_{x+12}
				$c_{1,1}$	$c_{1,2}$	$c_{1,3}$						
							$c_{2,1}$	c_{22}	$c_{2,3}$			
										$c_{3,1}$	$c_{3,2}$	$c_{3,3}$

Table 2.6 shows that all the cells were forwarded out the VOMQ in the same pattern they arrived and one cell each k time slots because the COM connects to the OM once each k time slots.

Also, let us assume that the first cell of a flow at the L_{CIM} arrives at least one or more time slots before the configuration of the COM allows forwarding the cell to its destination OM. Thus, cells may depart in the following or a few time slot after its

arrival. A cell then may wait up to $k-1$ time slots for the designated interconnection to take place before being forwarded to the OM.

Given k flows, with their τth cells being $c_{1,\tau}$ to $c_{k,\tau}$, the arrival time of the first arriving cell $c_{1,\tau}$ is:

$$t_{a_{1,\tau}} = t_x \tag{2.54}$$

The number of cells at the VOQ, $N_1(c_{y,\tau})$, upon the arrival of $c_{1,\tau}$ is:

$$N_1(c_{1,\tau}) = 0 \tag{2.55}$$

This condition holds because there is no cell at the VOQ when $c_{1,\tau}$ arrives. Because of (2.55), the queuing delay at the VOQ of $c_{1,\tau}$ is:

$$q_{1_{1,\tau}} = 0 \tag{2.56}$$

The departure time of a cell $c_{y,\tau}$ from the VOQ is:

$$d_{1_{y,\tau}} = t_{a_{y,\tau}} + q_{1_{y,\tau}} \tag{2.57}$$

Using (2.54) to (2.57), the departure time of $c_{1,\tau}$ from the VOQ is:

$$d_{1_{1,\tau}} = t_x + 1 \tag{2.58}$$

Upon arriving at the VOMQ, $c_{1,\tau}$ finds no cell ahead of it. Thus, the number of cells at the VOMQ, $N_2(c_{1,\tau})$, upon the arrival of $c_{1,\tau}$ is:

$$N_2(c_{1,\tau}) = 0 \tag{2.59}$$

Based on the considered traffic pattern, $c_{1,\tau}$ is stored in the VOMQ for additional $k-1$ time slots. Therefore,

$$q_{2_{1,\tau}} = k - 1 \tag{2.60}$$

The departure time of a cell $c_{y,\tau}$ from the VOMQ is:

$$d_{2_{y,\tau}} = d_{1_{y,\tau}} + q_{2_{y,\tau}} \tag{2.61}$$

Using (2.58), (2.60), and (2.61), the departure time of $c_{1,\tau}$ from the VOMQ is:

$$d_{2_{1,\tau}} = t_x + k \tag{2.62}$$

Let us consider now another cell from the same flow, $c_{1,\tau+\theta}$, where $0 < \theta < k$, with

$$t_{a_{1,\tau+\theta}} = t_x + \theta \tag{2.63}$$

Upon the arrival of $c_{1,\tau+\theta}$, there is no cell at the VOQ, or:

$$N_1(c_{1,\tau+\theta}) = 0 \tag{2.64}$$

Because of (2.59) and (2.64), the queuing delay at the VOQ for $c_{1,\tau+\theta}$ is:

$$q_{1_{1,\tau+\theta}} = 0 \tag{2.65}$$

Using (2.57), (2.63), and (2.65), the departure time of $c_{1,\tau+\theta}$ from the VOQ is:

$$d_{1_{1,\tau+\theta}} = t_x + \theta + 1 \tag{2.66}$$

Upon arriving at the VOMQ, $c_{1,\tau+\theta}$ finds no cell ahead of it, or:

$$N_2(c_{1,\tau+\theta}) = 0 \tag{2.67}$$

Because of the considered traffic, $c_{1,\tau+\theta}$ is queued extra $k-1$ time slots at the VOMQ, hence:

$$q_{2_{1,\tau+\theta}} = k - 1 \tag{2.68}$$

Using (2.61), and (2.66) to (2.68),

$$d_{2_{1,\tau+\theta}} = t_x + k + \theta \tag{2.69}$$

Using (2.62), therefore,

$$d_{2_{1,\tau+\theta}} = d_{2_{1,\tau}} + \theta \tag{2.70}$$

In general, for $c_{z,\tau}$, where $1 < z \leq k$, the arrival time is

$$t_{a_{z,\tau}} = t_x + (z - 1) \tag{2.71}$$

and upon the arrival of $c_{z,\tau}$ in the VOQ, there is no cell:

$$N_1(c_{z,\tau}) = 0 \tag{2.72}$$

With (2.72),

$$q_{1_{z,\tau}} = 0 \tag{2.73}$$

Using (2.57), (2.71), and (2.73),

$$d_{1_{z,\tau}} = t_x + z \tag{2.74}$$

However, upon arriving in the VOMQ, $c_{z,\tau}$ finds δ cells ahead of it, or:

$$N_2(c_{z,\tau}) = \delta \tag{2.75}$$

$$\delta = z - 1 \tag{2.76}$$

where $0 < \delta < k$

$$q_{2_{z,\tau}} = q_{H_{z,\tau}} + (\delta - 1)k + k \tag{2.77}$$

$q_{H_{z,\tau}}$ is the delay from the HoL cell in the VOMQ on $c_{z,\tau}$. $(\delta - 1)k$ is the delay generated from the other $(\delta - 1)$ cells ahead of $c_{z,\tau}$ in the VOMQ. The extra k time slots is the delay $c_{z,\tau}$ experiences as it waits for the configuration pattern to repeat after the last cell ahead of it is forwarded to the OM. where

$$d_{2_{1,\tau}} = d_{1_{z,\tau}} + q_{H_{z,\tau}} \tag{2.78}$$

Using (2.61), (2.77), and (2.78), the departure time of $c_{z,\tau}$ from the VOMQ is:

$$d_{2_{z,\tau}} = d_{2_{1,\tau}} + \delta k \tag{2.79}$$

Using (2.62) and (2.76), then:

$$d_{2_{z,\tau}} = t_x + zk \tag{2.80}$$

Let us now consider any other cell from flow z, $c_{z,\tau+\theta}$, where $0 < \theta < k$. The time of arrival of the cell $c_{z,\tau+\theta}$ is:

$$t_{a_{z,\tau+\theta}} = t_x + (z - 1) + \theta \tag{2.81}$$

Upon the arrival of $c_{z,\tau+\theta}$, there could be zero or more at the VOQ, hence:

$$N_1(c_{z,\tau+\theta}) = \gamma \tag{2.82}$$

where γ is the number of cells at the VOQ upon the arrival of $c_{z,\tau+\theta}$ and $0 \leq \gamma < k$. Using (2.75) and (2.82), then:

$$q_{1_{z,\tau+\theta}} = \begin{cases} \delta k & for\ \gamma = 0 \\ \delta k + \sum_{\sigma=1}^{\theta-1} q_{1_{z,\tau+\sigma}} & for\ \gamma > 0 \end{cases} \tag{2.83}$$

where

$$\sum_{\sigma=1}^{\theta-1} q_{1_{z,\tau+\sigma}}$$

is the delay generated from the γ cells ahead of $c_{z,\tau+\theta}$ at the VOQ. Let

$$\gamma_q = \sum_{\sigma=1}^{\theta-1} q_{1_{z,\tau+\sigma}} \tag{2.84}$$

Using (2.57), (2.81), (2.83), and (2.84), then:

$$d_{1_{z,\tau+\theta}} = \begin{cases} t_x + (z-1) + \theta + \delta k & for\ \gamma = 0 \\ t_x + (z-1) + \theta + \delta k + \gamma_q & for\ \gamma > 0 \end{cases} \tag{2.85}$$

The queuing delay of $c_{z,\tau+\theta}$ at the VOMQ is equal to (2.77). Therefore, using (2.61), (2.77), and (2.85), the departure time of $c_{z,\tau+\theta}$ from the VOMQ is:

$$d_{2_{z,\tau+\theta}} = \begin{cases} d_{2_{1,\tau+\theta}} + \delta k & for\ \gamma = 0 \\ d_{2_{1,\tau+\theta}} + \delta k + \gamma_q & for\ \gamma > 0 \end{cases} \tag{2.86}$$

Using (2.70) and (2.76), then:

$$d_{2_{z,\tau+\theta}} = \begin{cases} d_{2_{1,\tau}} + (z-1)k + \theta & for\ \gamma = 0 \\ d_{2_{1,\tau}} + (z-1)k + \theta + \gamma_q & for\ \gamma > 0 \end{cases} \tag{2.87}$$

Using (2.62), then:

$$d_{2_{z,\tau+\theta}} = \begin{cases} t_x + zk + \theta & for\ \gamma = 0 \\ t_x + zk + \theta + \gamma_q & for\ \gamma > 0 \end{cases} \tag{2.88}$$

From (2.70),

$$d_{2_{1,\tau+\theta}} - d_{2_{1,\tau}} = \theta \tag{2.89}$$

Using (2.80), gives:

$$d_{2_{z,\tau+\theta}} - d_{2_{z,\tau}} = \begin{cases} \theta & for\ \gamma = 0 \\ \theta + \gamma_q & for\ \gamma > 0 \end{cases} \tag{2.90}$$

The difference between the departure times of any two cells of a flow from VOMQ is a function of θ, which is the arrival time difference of the two cells. Therefore, cells of a flow are forwarded to the OM in the same order they arrived.

■

Lemma 3. *For any number of flows traversing the LBC switch, the cells of each flow arrive and are cleared at the output port (OP) in the same order the cells arrived at the input port (IP).*

In our discussion of this lemma, let us consider the following traffic scenario: The switch has cells from only two flows, each arriving in a different IM (and therefore IP) and both of them are destined to the same OP. In each flow, cells arrive back-to-back, one at each time slot, and the first cell of both flows arrive at a time slot such that the configuration pattern of IM-CIM stage would not enable forwarding them to the COM immediately. With this condition, we analyze how these two flows are kept from affecting each other, and therefore, the sequence in which cells may depart the OP. This traffic scenario may present the greatest opportunity of experiencing out-of-sequence forwarding by any two cells of a flow as cells from these two flows interact at the CBs of the destination OP. Let us also consider empty queues as an initial condition.

Given flows y and z, where the first cells of y and z, $c_{y,\tau}$ and $c_{z,\tau}$, respectively, arrive at their respective VOQs at time slot t_x and the θth cells, $c_{y,\tau+\theta}$ and $c_{z,\tau+\theta}$ $\forall$ $\theta \geq 1$, arrive at time slot $t_x + \theta$. Therefore, according to this lemma $c_{y,\tau}$ and $c_{z,\tau}$ must be forwarded and cleared from the output port $OP(j,d)$ before $c_{y,\tau+\theta}$ and $c_{z,\tau+\theta}$, respectively.

Proof:

We analyze the departure time of the cells $c_{y,\tau}$ and $c_{z,\tau}$ from the CBs. The arrival times for cells $c_{y,\tau}$ and $c_{z,\tau}$ is:

$$t_{a_{y,\tau}} = t_{a_{z,\tau}} = t_x \tag{2.91}$$

Upon arriving in the VOQ, $c_{y,\tau}$ and $c_{z,\tau}$ are placed as HoL cells. Because there are no backlogged cells, hence:

$$N_1(c_{y,\tau}) = 0 \tag{2.92}$$

and

$$N_1(c_{z,\tau}) = 0 \tag{2.93}$$

Using (2.92) and (2.93), the queuing delays of $c_{y,\tau}$ and $c_{z,\tau}$ at the VOQ are:

$$q_1 c_{y,\tau} = 0 \tag{2.94}$$

and

$$q_1 c_{z,\tau} = 0 \tag{2.95}$$

Using (2.57), (2.91), and (2.94) the departure time for $c_{y,\tau}$ from the VOQ is:

$$d_{1y,\tau} = t_x + 1 \tag{2.96}$$

Using (2.57), (2.91), and (2.95) the departure time for $c_{z,\tau}$ from the VOQ is:

$$d_{1z,\tau} = t_x + 1 \tag{2.97}$$

Thus, $c_{y,\tau}$ and $c_{z,\tau}$ are forwarded to the same CIM (so that these two cells would share the same CB) and stored in their respective VOMQ. Because the VOMQs are empty at the time the two cells arrive, hence:

$$N_2(c_{y,\tau}) = 0 \tag{2.98}$$

and

$$N_2(c_{z,\tau}) = 0 \tag{2.99}$$

Based on the adopted traffic scenario, $c_{y,\tau}$ and $c_{z,\tau}$ are held at the VOMQ for β_1 and β_2 time slots, respectively, before the configuration pattern enables forwarding them to their destination OM. Here, $1 \leq \beta_1 < k$ and $1 \leq \beta_2 < k$. Hence, the queuing delay of $c_{y,\tau}$ at the VOMQ is:

$$q_{2_{y,\tau}} = \beta_1 \tag{2.100}$$

The queuing delay of $c_{z,\tau}$ at the VOMQ is:

$$q_{2_{z,\tau}} = \beta_2 \tag{2.101}$$

Assuming $\beta_1 < \beta_2$, hence $c_{y,\tau}$ would be forwarded to the destination OM before $c_{z,\tau}$. From (2.61), (2.96), and (2.100), the departure time of $c_{y,\tau}$ from the VOMQs is:

$$d_{2y,\tau} = t_x + 1 + \beta_1 \tag{2.102}$$

From (2.61), (2.97), and (2.101), the departure time of $c_{z,\tau}$ from the VOMQs is:

$$d_{2z,\tau} = t_x + 1 + \beta_2 \tag{2.103}$$

When $c_{y,\tau}$ and $c_{z,\tau}$ arrive at the OM, they are stored at CBs before being forwarded to the output port.

Let us now consider $c_{y,\tau+1}$ and $c_{z,\tau+1}$, which arrive at time slot $t_x + 1$, hence:

$$t_{a_{y,\tau+1}} = t_{a_{z,\tau+1}} = t_x + 1 \tag{2.104}$$

Because there are no cells at the VOQ upon the arrival of $c_{y,\tau+1}$ and $c_{z,\tau+1}$, then:

$$N_1 c_{y,\tau+1} = 0 \tag{2.105}$$

and

$$N_1 c_{z,\tau+1} = 0 \tag{2.106}$$

With (2.98) and (2.105), the queuing delay of $c_{y,\tau+1}$ at the VOQ is:

$$q_{1y,\tau+1} = 0 \tag{2.107}$$

With (2.99) and (2.106), the queuing delay of $c_{z,\tau+1}$ at the VOQ is:

$$q_{1z,\tau+1} = 0 \tag{2.108}$$

Using (2.57), (2.104), and (2.107), the departure time of $c_{y,\tau+1}$ from the VOQ is:

$$d_{1y,\tau+1} = t_x + 2 \tag{2.109}$$

Using (2.57), (2.104), and (2.108), the departure time of $c_{z,\tau+1}$ from the VOQ is:

$$d_{1z,\tau+1} = t_x + 2 \tag{2.110}$$

$c_{y,\tau+1}$ and $c_{z,\tau+1}$ are forwarded to the same CIM and stored in their respective VOMQs. Based on the traffic scenario $c_{y,\tau+1}$ and $c_{z,\tau+1}$ are also stored for β_1 and β_2 time slots, respectively, at the VOMQs before the configuration pattern of the COM enables forwarding them to the destination OM. Hence, the queuing delay of $c_{y,\tau+1}$ and $c_{z,\tau+1}$ at the VOMQ are equal to (2.100) and (2.101), respectively. From (2.61), (2.100), and (2.109), the departure time of $c_{y,\tau+1}$ from the VOMQ is:

$$d_{2y,\tau+1} = t_x + 2 + \beta_1 \tag{2.111}$$

From (2.61), (2.101), and (2.110), the departure time of $c_{z,\tau+1}$ from the VOMQ is:

$$d_{2z,\tau+1} = t_x + 2 + \beta_2 \tag{2.112}$$

Next, we analyze the departure time of the cells from the output port. Because $d_{2y,\tau+1} > d_{2y,\tau}$ and $d_{2z,\tau+1} > d_{2z,\tau}$, this means that $c_{y,\tau}$ and $c_{z,\tau}$ arrive at the output module before $c_{y,\tau+1}$ and $c_{y,\tau+1}$, respectively. With the CB initially empty based on the initial condition, then:

$$N_3 c_{y,\tau} = 0 \tag{2.113}$$

With $d_{2z,\tau} > d_{2y,\tau}$, hence:

$$N_3 c_{z,\tau} = 0 \tag{2.114}$$

With (2.113) and (2.114), the queuing delays of $c_{y,\tau}$ and $c_{z,\tau}$ at the CB are:

$$q_{3y,\tau} = 0 \tag{2.115}$$

and

$$q_{3z,\tau} = 0 \tag{2.116}$$

The queuing delay of $c_{y,\tau+1}$ and $c_{z,\tau+1}$ at the CB are equal to (2.115) and (2.116). The departure time of a cell $c_{c,\tau}$ from the CB is:

$$d_{3_{c,\tau}} = d_{3_{c,\tau}} + q_{3_{c,\tau}} \tag{2.117}$$

Therefore, using (2.102), (2.115), and (2.117), the departure time of $c_{y,\tau}$ from the output port is:

$$d_{3y,\tau} = t_x + 2 + \beta_1$$

Using (2.111), (2.115), and (2.117), the departure time of $c_{y,\tau+1}$ from the output port is:

$$d_{3y,\tau+1} = t_x + 3 + \beta_1$$

Using (2.103), (2.116), and (2.117), the departure time of $c_{z,\tau}$ from the output port is:

$$d_{3z,\tau} = t_x + 2 + \beta_2$$

Using (2.112), (2.116), and (2.117), the departure time of $c_{z,\tau+1}$ from the output port is:

$$d_{3z,\tau+1} = t_x + 3 + \beta_2$$

Therefore, with $d_{3y,\tau+1} > d_{3y,\tau}$ and $d_{3z,\tau+1} > d_{3z,\tau}$, $c_{y,\tau}$ and $c_{z,\tau}$ would depart the output port before $c_{y,\tau+1}$ and $c_{z,\tau+1}$, respectively. Note that for $N_1(c_{y,\tau}) > 0$, $\delta > 0$, such that the cells from the same flow are forwarded with larger time separation from each other, and there are fewer chances that they will be at the CBs at the same time slot. Therefore, this property, as described by this lemma, applies to any two cells of a flow.

■

This completes the proof of Theorem 1.

■

2.5 Performance Analysis

We evaluated the performance of the LBC switch through computer simulation under both uniform and nonuniform traffic models. We also compared the performance of the proposed switch with that of an output-queued (OQ) switch, a high-performing Memory-Memory-Memory Clos-network (MMM) switch, and an MMM switch with extended memory (MM^{e}M). The MMM switch uses forwarding arbitration schemes to select cells from the buffers in the previous stage modules and is agnostic to cell

sequence, therefore delivering high switching performance. We considered switches with sizes $N = \{64, 256\}$.

2.5.1 Uniform Traffic

We evaluated the LBC, OQ, MMM, and MM^{e}M switches under uniform traffic with Bernoulli and bursty arrivals. Figures 2.5 and 2.6 show the average delay under uniform Bernoulli traffic arrivals for $N = 64$ and $N = 256$, respectively. The results in the figures show that the LBC switch achieves 100% throughput under uniform traffic with Bernoulli arrivals, indicated by the finite and moderate average queuing delay. The high throughput performance by the proposed switch is the result of using an efficient load-balancing process in the IM-CIM stage. However, this high performance is expected under this traffic pattern as the distribution of the incoming traffic is already uniformly distributed.

Figure 2.5 shows that the LBC switch experiences a similar delay as the MM^{e}M switch at high input load. Figure 2.6 shows that the LBC switch experiences a slightly higher average delay than the OQ switch. This additional delay in the LBC switch is caused by having cells wait in the VOMQs until a configuration that allows forwarding the cells to their destination output modules takes place. Because MM^{e}M requires an excessive amount of memory to implement the extended set of queues, the measurement of average cell delay cannot be measured for N=256 by our simulators. This figure also shows that the LBC switch achieves a lower average delay than the MMM switch with an input load of 0.95 and larger.

Uniform bursty traffic is modeled as an ON-OFF Markov modulated process, with the average duration of the ON period set as the average burst length, l, with $l = \{10, 30\}$ cells. Figures 2.7 and 2.8 show the average delay under uniform traffic with bursty arrivals for average burst length of 10 and 30 cells, respectively, for switches with N=256. The results show that the LBC switch achieves 100% throughput under

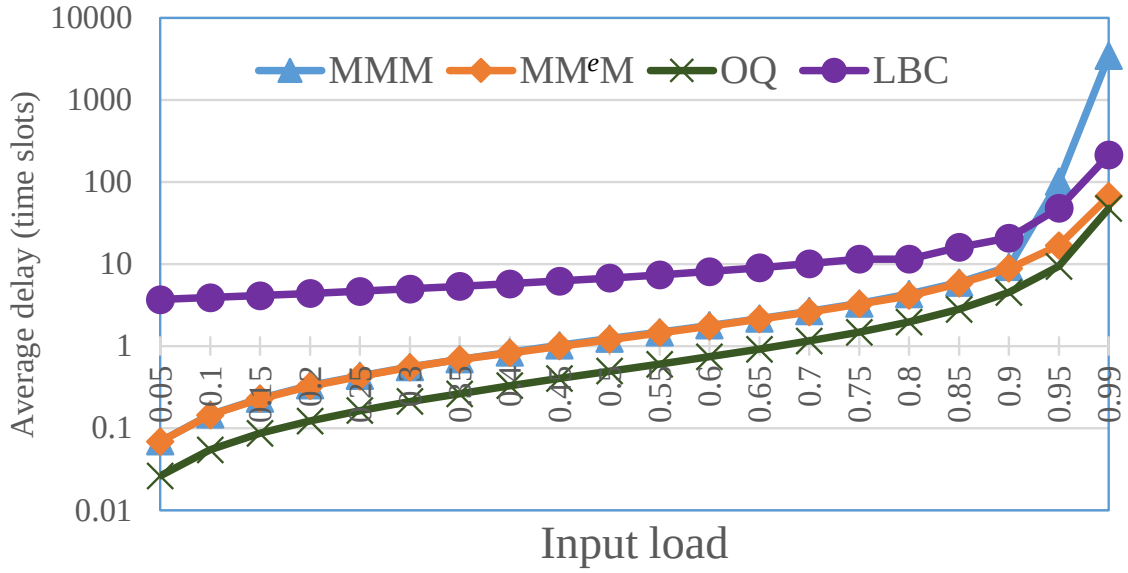


Figure 2.5 Average queueing delay under uniform traffic for N=64.

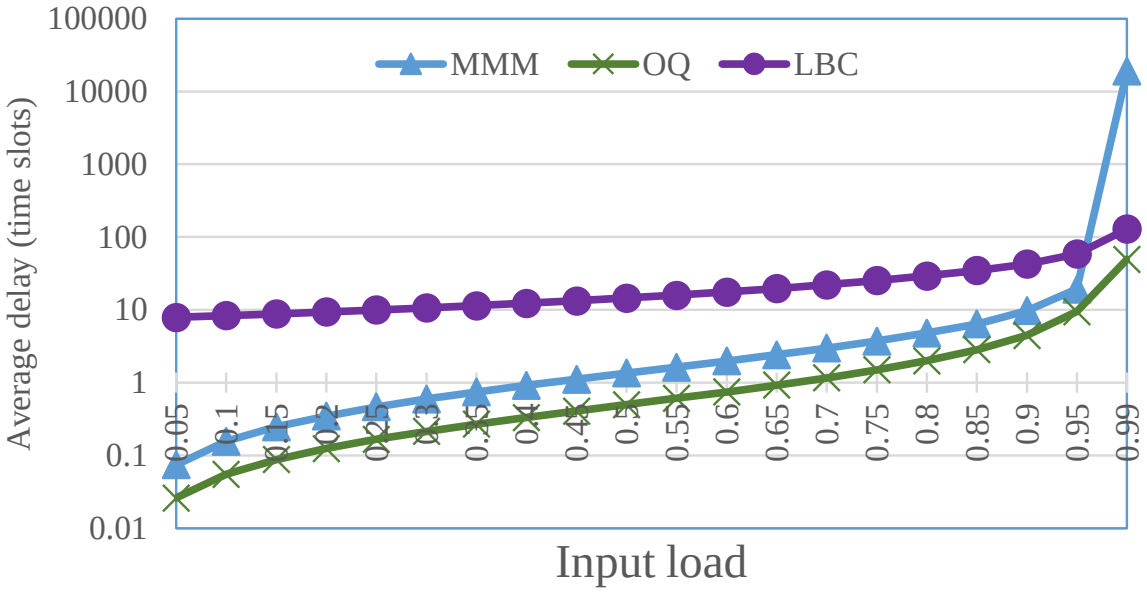


Figure 2.6 Average queueing delay under uniform traffic for N=256.

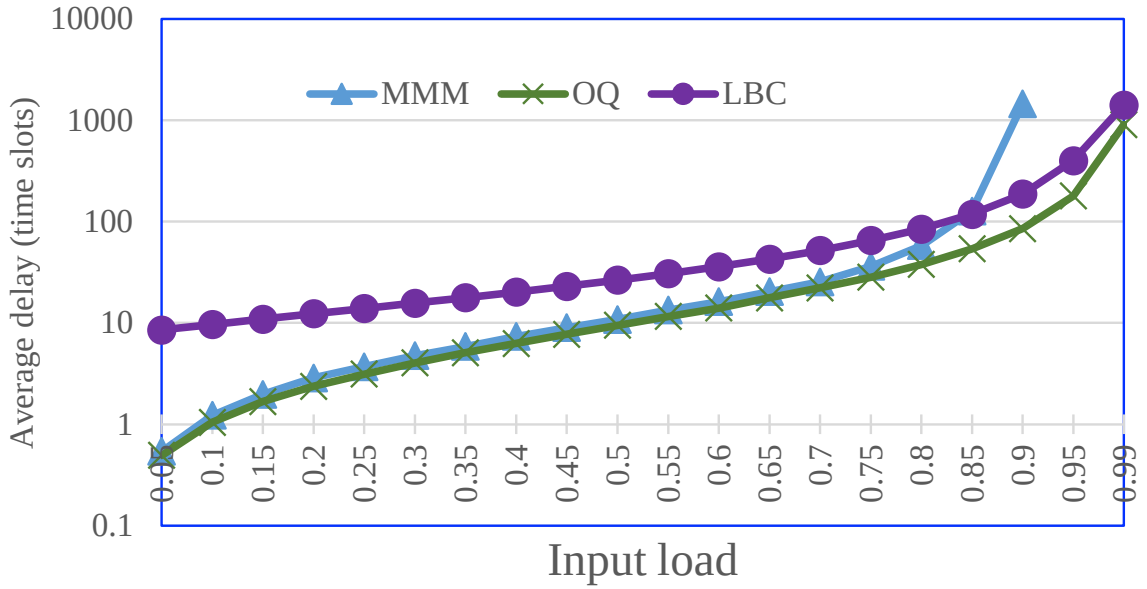


Figure 2.7 Average queuing delay under uniform bursty traffic with average burst length l=10 for N=256.

bursty uniform traffic. In contrast, the MMM switch has a throughput of 0.8 and 0.75 for an average burst length of 10 and 30 cells, respectively. Therefore, the LBC switch achieves a performance closer to that of the OQ switch. There is a very small difference in the delay of the LBC. From this graph, we also observe that the LBC switch achieves 100% throughput under bursty uniform traffic.

The uniform distributed nature of the traffic and the load-balancing stages help to achieve this high throughput and low queueing delay. The slightly larger average queueing delay of the LBC switch for very small input loads is caused by the predetermined and cyclic configuration of the bufferless modules as some cells wait for a few time slots to be forwarded and this is irrespective of the switch size. Nevertheless, these two figures show that the queueing delay difference between the LBC and the OQ switch is not significant for large input loads.

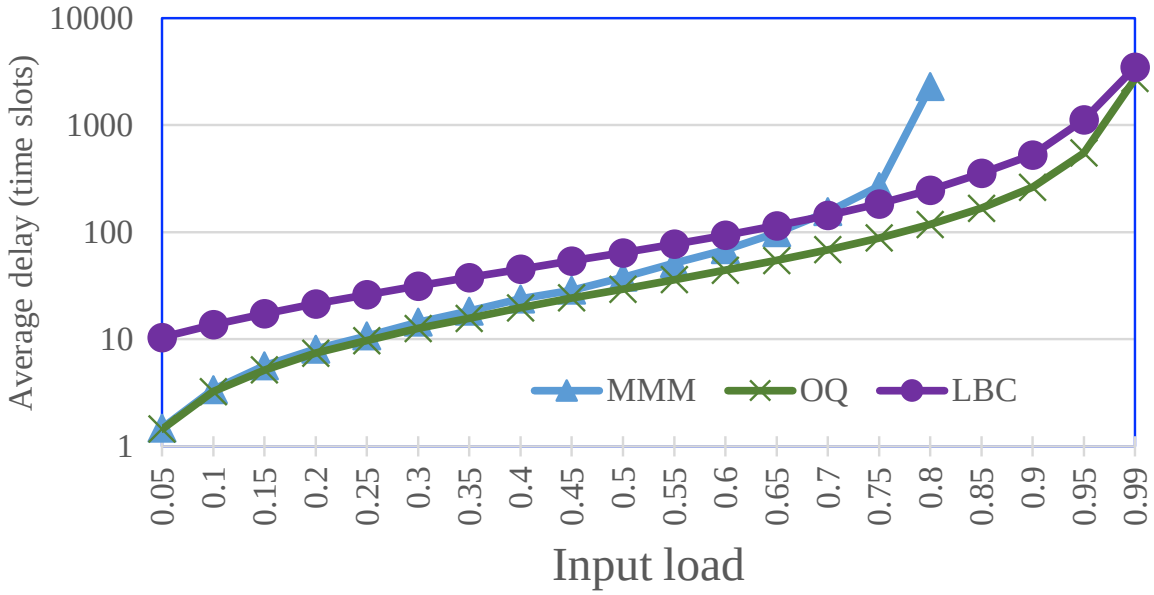


Figure 2.8 Average queuing delay under uniform bursty traffic with average burst length l=30 for N=256.

2.5.2 Nonuniform Traffic

We also compared the performance of the proposed LBC switch with the MMM, MM^{e}M, and OQ switches under unbalanced [72, 74] and hot-spot patterns as nonuniform traffic. The unbalanced traffic can be modeled using an unbalanced probability ω to indicate the load variances for different flows. Consider input port $IP(i,s)$ and output port $OP_(j,d)$ of the LBC switch, the traffic load is determined by

$$\rho_{i,s,j,d} = \begin{cases} \rho(\omega + \dfrac{1-\omega}{N}), & \text{if } i = j \text{ and } s = d, \\ \rho\dfrac{1-\omega}{N}, & \text{otherwise} \end{cases} \tag{2.118}$$

where ρ is the traffic load for input $IP(i,s)$ and ω is the unbalanced probability. When ω=0, the input traffic is uniformly distributed and when ω=1, the input traffic is completely directional; traffic from $IP(i,s)$ is destined for $OP(j,d)$.

The simulation results show that the throughput of the LBC switch is 100% under this traffic pattern for all values of ω, matching those of MMM and MM^{e}M

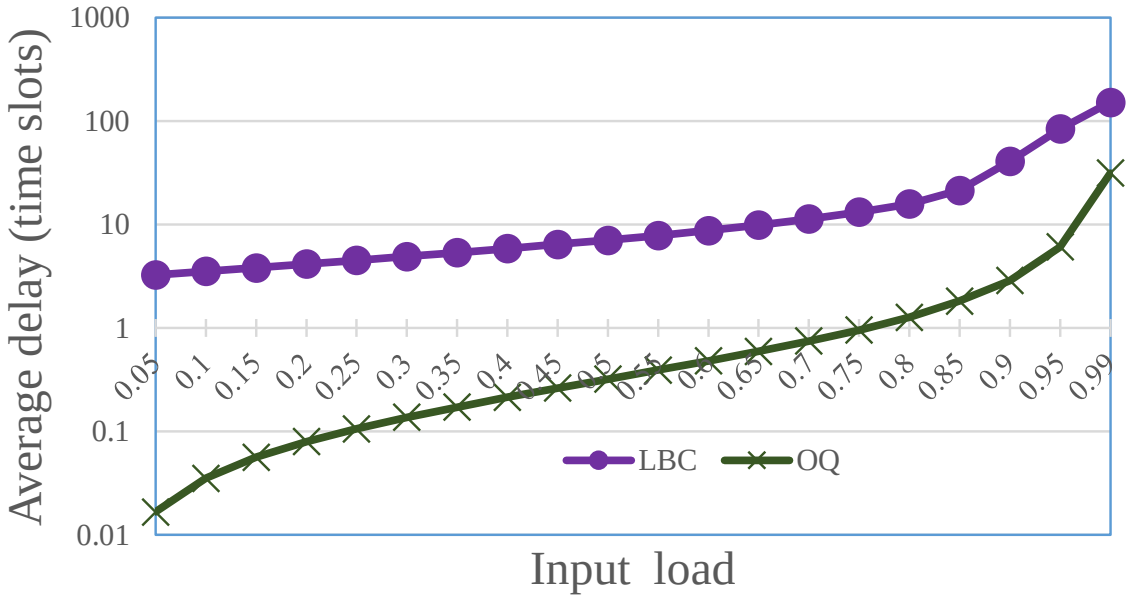


Figure 2.9 Average queuing delay under unbalanced traffic with $w = 0.6$ for N=256.

switches, which are also known to achieve high throughput but neglect in-sequence forwarding. It has been shown that many switches do not achieve high throughput when w is around 0.6 [72]. Therefore, we measured the average delay of the LBC switch under this traffic pattern for ω=0.6, as shown in Figure 2.9, and compared with the OQ switch as this switch is well-known to achieve 100% throughput. As the figure shows, the average delay of the LBC switch is comparable to that of an OQ switch. The load-balancing stage of the LBC switch distributes the traffic uniformly throughout the switch.

We compared the performance of the proposed LBC switch with the MMM, MM^{e}M, and OQ switches under hot-spot traffic [64]. Hot-spot traffic occurs when all IPs send most or all traffic to one OP. Consider input port $IP(i, s)$ and output port $OP(j, d)$ of the LBC switch, the traffic load is determined by

$$\rho_{i,s,j,d} = \begin{cases} \rho(\frac{1}{N}), & \text{for } jm + d = h, \\ 0, & \text{otherwise} \end{cases} \tag{2.119}$$

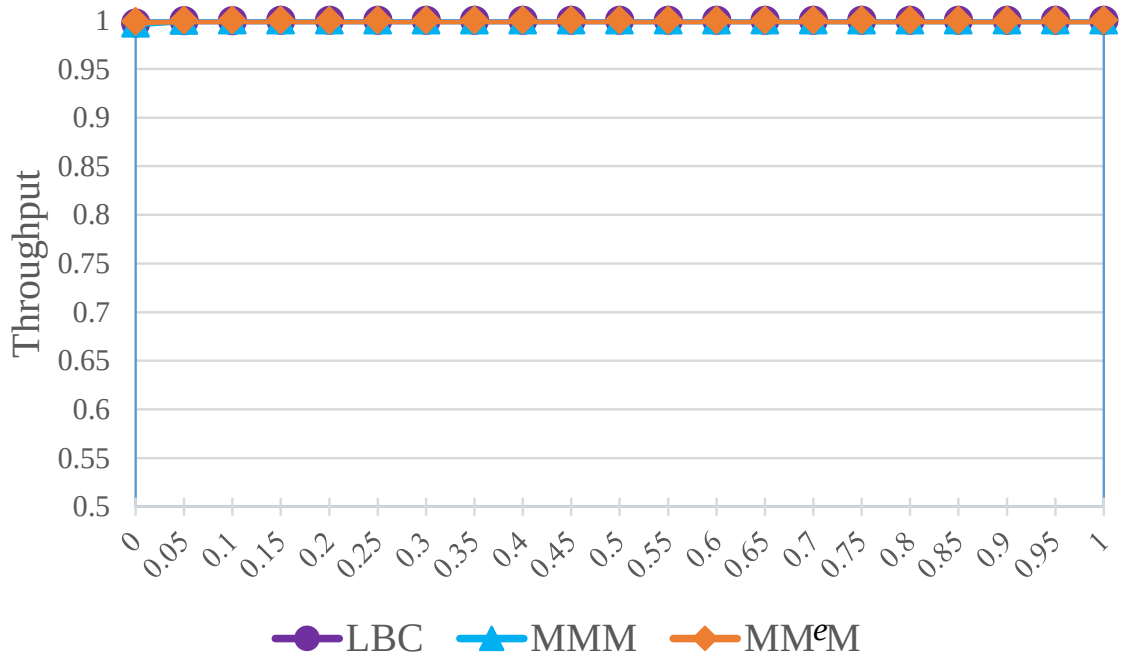


Figure 2.10 Throughput under hot-spot traffic for N=256.

where h is the hot-spot OP and $1 \leq h \leq N$.

Our simulation shows that the LBC switch as well as the MMM and MM^{e}M switches achieve 100% throughput under admissible hot-spot traffic.

Figure 2.10 shows the measured average delay of the LBC switch under this traffic pattern and that of an OQ switch. The figure shows that the average delay of the LBC switch is comparable to that of an OQ switch. This is as a result of effective load-balancing at the IMs, CIMs, and COMs of the multiple flows coming from different inputs.

In addition to the analysis presented in Section 2.2.6, we also simulated the LBC switch under two new traffic patterns, which we believe may stress the occupancy of CBs and therefore increase the likelihood of occurrence of HoL blocking conditions. The traffic patterns are: a) k flows from IPs at different IMs, each arriving at a rate of $\frac{1}{k}$ for admissibility, are forwarded to all OPs at one OM. The source IPs of the flows are selected such that they share VOMQs; $i = s$ or $IP(0,0), IP(1,1), \cdots, IP(k-1, n-1)$.

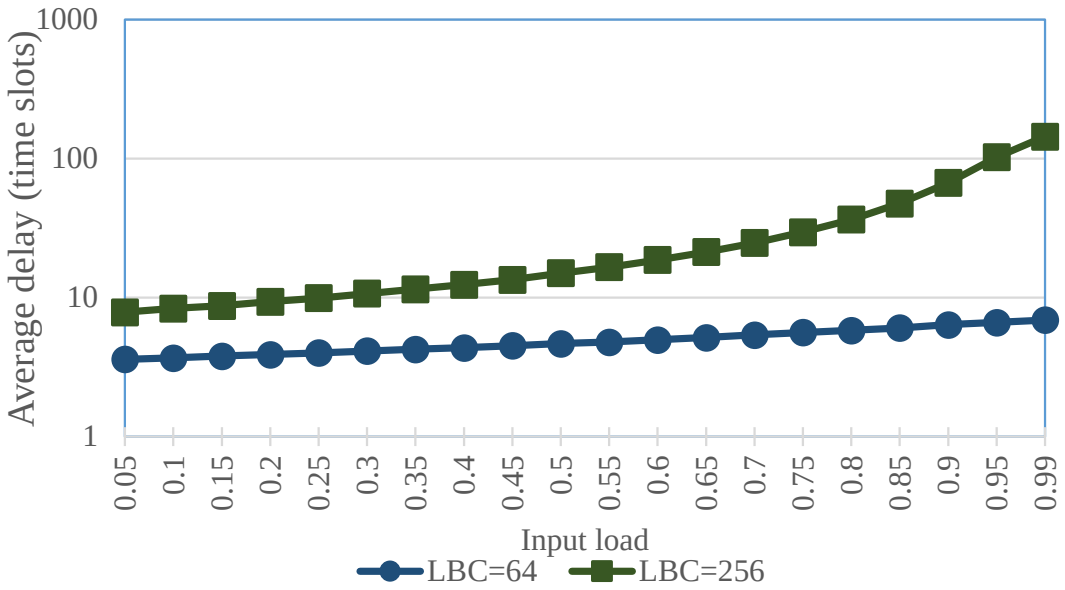


Figure 2.11 Average queueing delay of LBC switch under k flows from k IMs to all OPs in an OM

b) Each IP at an IM forwards cells at rate $\frac{1}{k}$ to each OP at an OM (e.g., $i = j$). Each OP in the destination OM receives traffic from all IPs of one IM. VOMQs are also shared by different flows. Figures 2.11 and 2.12 show the average delay under the first and second traffic patterns presented above, respectively. The results in the figures show that LBC experiences a finite and moderate average queuing delay, which implies that LBC achieves 100% throughput under both traffic patterns. We also measured the average CB length and this length does not grow more than one cell, indicating that no CB gets congested. This result is obtained because the load-balancing mechanism spreads a flow to different VOMQs.

2.6 Conclusions

We have introduced a configuration scheme for a split-central-buffered load-balancing Clos-network switch and a mechanism that forwards cells in sequence for this switch. To effectively perform load balancing, the switch has virtual output module queues

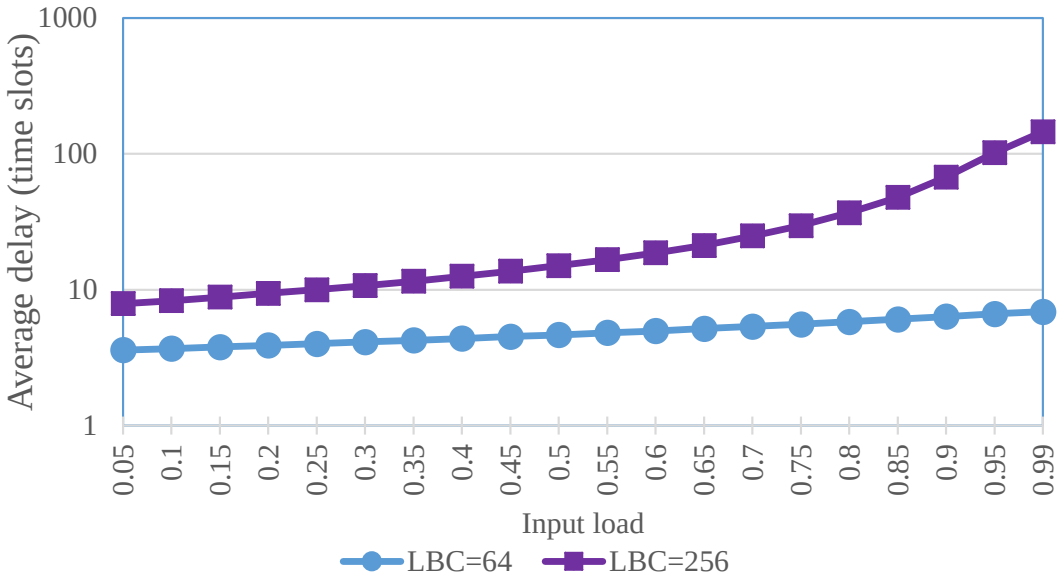


Figure 2.12 Average queueing delay of LBC switch under hot-spot per-Module traffic

between these two central stages. With the split central module, the switch comprises four stages, named IM, CIM, COM, and OM. The IM, CIM, and COM stages are bufferless crossbars, while the OMs is a buffered one. All the bufferless modules follow a pre-deterministic configuration while the OM follows a round-robin sequence to forward cells from the CB to the output ports. Therefore, the switch does not have to perform matching in any stage despite having bufferless modules, and the configuration complexity of the switch is minimum, making it comparable to that of MMM switches. We introduced an in-sequence mechanism that operates at the inputs of the LBC switch to avoid out-of-sequence forwarding caused by the central buffers. We modeled and analyzed the operations that each of the stages effects on the incoming traffic to obtain the loads seen by the output ports. We showed that for admissible independent and identically distributed traffic, the switch achieves 100% throughput. Unlike the existing switching architectures discussed in Section 1, LBC achieves high performance, configuration simplicity, and in-sequence service

without memory speedup and central module expansion. In addition, we analyzed the operation of the forwarding mechanism and demonstrated that cells are forwarded in sequence. We showed, through computer simulation, that for all tested traffic, the switch achieved 100% throughput for uniform and nonuniform traffic distributions.

CHAPTER 3

TRIDENT: A LOAD-BALANCING CLOS-NETWORK PACKET SWITCH WITH QUEUES BETWEEN INPUT AND CENTRAL STAGES AND IN-ORDER FORWARDING

3.1 Introduction

In this chapter we propose a load-balancing Clos-network switch that has buffers placed between the IMs and CMs. Furthermore, we use OMs implemented with buffered crossbars with per-flow queues. The switch is called ThRee-stage Clos swItch with queues at the middle stage and DEtermiNisTic scheduling (TRIDENT). This switch uses predetermined and periodic interconnection patterns for the configuration of IMs and CMs. The incoming traffic is load-balanced by IMs and routed by CMs and OMs. The result is a switch that attains high throughput under admissible traffic with independent and identical distribution (i.i.d.) and uses a configuration scheme with $O(1)$ complexity. The switch also adopts an in-sequence forwarding mechanism at the input ports and output modules to keep cells in sequence.

We analyze the performance of the proposed switch by modeling the effect of each stage on the traffic passing through the switch. In addition, we study the performance of the switch through traffic analysis and by computer simulation. We show that the switch attains 100% throughput under several admissible traffic models, including traffic with uniform and nonuniform distributions, and demonstrate that the switch forwards cells to the output ports in sequence. This high switching performance is achieved without resorting to speedup nor switch expansion.

The remainder of this chapter is organized as follows: In Section 3.2, we introduce the TRIDENT switch. In Section 3.3, we analyze the performance of TRIDENT by modeling the effect of each stage on the traffic passing through the

switch. We also present the throughput analysis of TRIDENT. In Section 3.4, we present a proof of the in-sequence forwarding property of TRIDENT. In Section 3.5, we present a simulation study on the performance of TRIDENT by computer simulation. In Section 3.6, we present our conclusions.

3.2 Switch Architecture

The TRIDENT switch has N inputs and N outputs, each denoted as $IP(i, s)$ and $OP(j, d)$, respectively, where $0 \leq i,\ j \leq k-1,\ 0 \leq s,\ d \leq n-1$, and $N = nk$. Figure 3.1 shows the architecture of TRIDENT. This switch has k $n \times m$ IMs, m $k \times k$ CMs, and k $m \times n$ OMs. Table 3.1 lists the notations used in the description of TRIDENT. In the remainder of this chapter, we set $n = k = m$ for symmetry and cost-effectiveness. The IMs and CMs are bufferless crossbars while the OMs are buffered ones. In order to preserve the staggered symmetry and in-order delivery [31], this switch uses a fixed and predetermined configuration sequence, and a reverse desynchronized configuration scheme in CMs. The staggered symmetry and in-order delivery refers to the fact that at time slot t, $IP(i, s)$ is connected to $CM(r)$, which in turn is connected to $OM(j)$. Then at the next time slot (t+1), $IP(i, s)$ is connected to $CM((r+1) \mod m)$, which in turn is connected to $OM(j)$. This property enables the configuration of IMs and CMs to be easily represented with a predetermined compound permutation that repeats every k time slots. This property also ensures that cells experience similar delay under uniform traffic, and the incorporation of the in-sequence mechanism enables preserving this delay under nonuniform traffic, as Section 3.4 shows.

The switch has virtual input-module output port queues (VIMOQs) between the IMs and CMs to store cells coming from $IM(i)$ and destined to $OP(j, d)$, and each queue is denoted as $VIMOQ(r, i, j, d)$. Each output of an IM is denoted as $L_I(i, r)$. Each output of a VIMOQ is connected to a CM, whose input and output are

denoted as $I_C(r,p)$ and $L_C(r,j)$, respectively. Each OP has Nk crosspoint buffers, each denoted as $CB(r,j,d,i,s)$ and designated for the traffic from each IP traversing different CMs to an OP. A flow control mechanism between CBs and VIMOQs is used to avoid buffer overflow and underflow [74].

Table 3.1 Notations used in the Description of the TRIDENT Switch

Term	Description
N	Number of input/output ports.
n	Number of input/output ports for each IM and OM.
m	Number of CMs.
k	Number of IMs and OMs, where $k = \frac{N}{n}$.
$IP(i,s)$	Input port s of $IM(i)$, where $0 \leq i \leq k-1, 0 \leq s \leq n-1$.
$IM(i)$	Input module i.
$CM(r)$	Central Input Module r, where $0 \leq r \leq m-1$.
$L_I(i,r)$	Output link of $IM(i)$ connected to $CM(r)$.
$I_C(r,p)$	Input port p of $CM(r)$.
$L_C(r,j)$	Output link of $CM(r)$ connected to $OM(j)$.
$VIMOQ(r,i,j,d)$	VIMOQ at input of CMs that stores cells destined to $OP(j,d)$.
$CB(r,j,d,i,s)$	Crosspoint buffer at $OM(j)$ that stores cells from $IP(i,s)$ going through $CM(r)$ and destined to $OP(j,d)$.
$OP(j,d)$	Output port d at $OM(j)$.

3.2.1 Module Configuration

The IMs are configured based on a predetermined sequence of k disjoint permutations, where one permutation is applied each time slot. We call a permutation disjoint from the set of permutations if an input-output pair is interconnected in one and only one of the permutations. Cells at the inputs of IMs are forwarded to the outputs of the IMs determined by the configuration at that time slot. A cell is then stored in the VIMOQ corresponding to its destination OP.

Similar to the IMs, CMs are configured based on a predetermined sequence of k disjoint permutations. Unlike IMs, CMs follow a desynchronized configuration, where each CM is configured with a different permutation and the configuration of

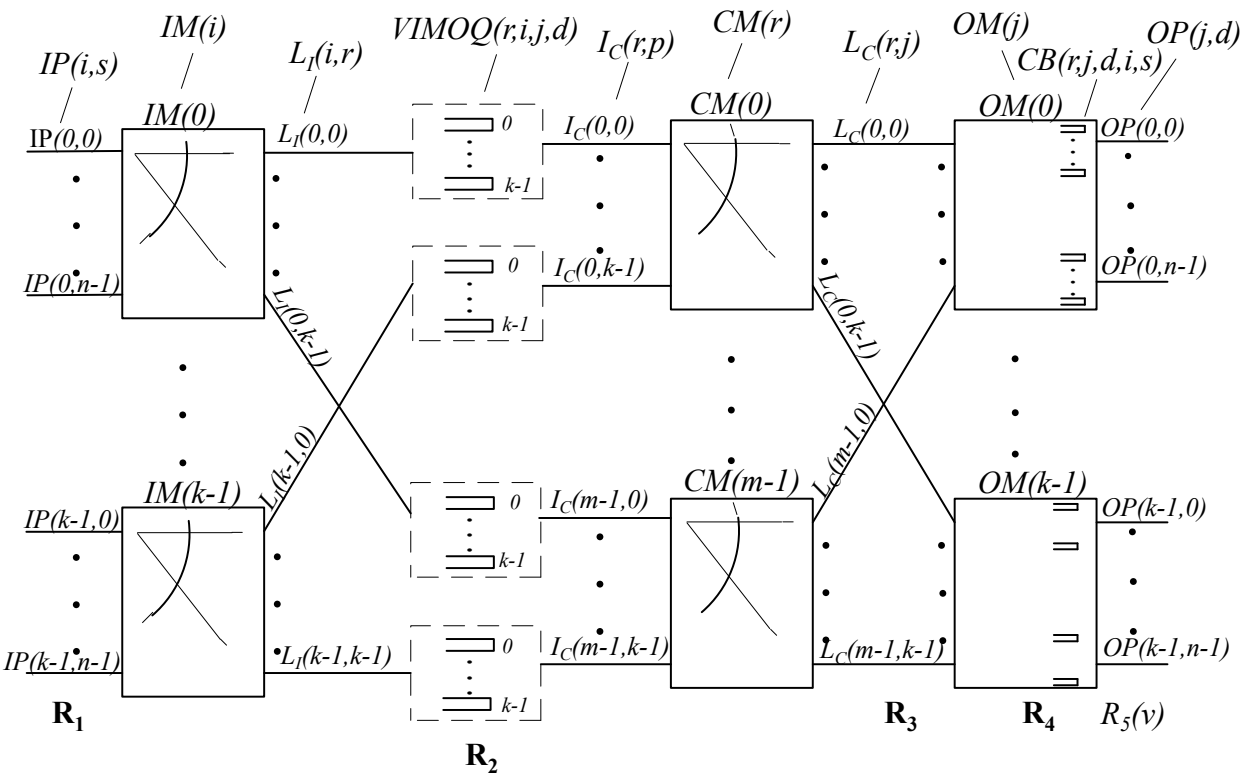


Figure 3.1 TRIDENT switch.

each CMs changes counter-clockwise each time slot. The Head-of-Line (HoL) cell at the VIMOQ destined to $OP(j,d)$ is forwarded to its destination when the input of the CM is connected to the input of the destined $OM(j)$. Else, the HoL cell waits until the required configuration takes place. The forwarded cell is queued at the CB of its destination OP once it arrives in the OM.

The configurations of the bufferless IMs and CMs are as follows. At time slot t, IM input $IP(i,s)$ is interconnected to IM output $L_I(i,r)$, as follows:

$$r = (s+t) \mod m \tag{3.1}$$

and each CM input $I_C(r,p)$ is interconnected to output $L_C(r,j)$ as follows:

$$j = (p-t+r) \mod k. \tag{3.2}$$

The use of CBs at an OP allows forwarding a cell from of a VIMOQ to its destined output without requiring port matching [48].

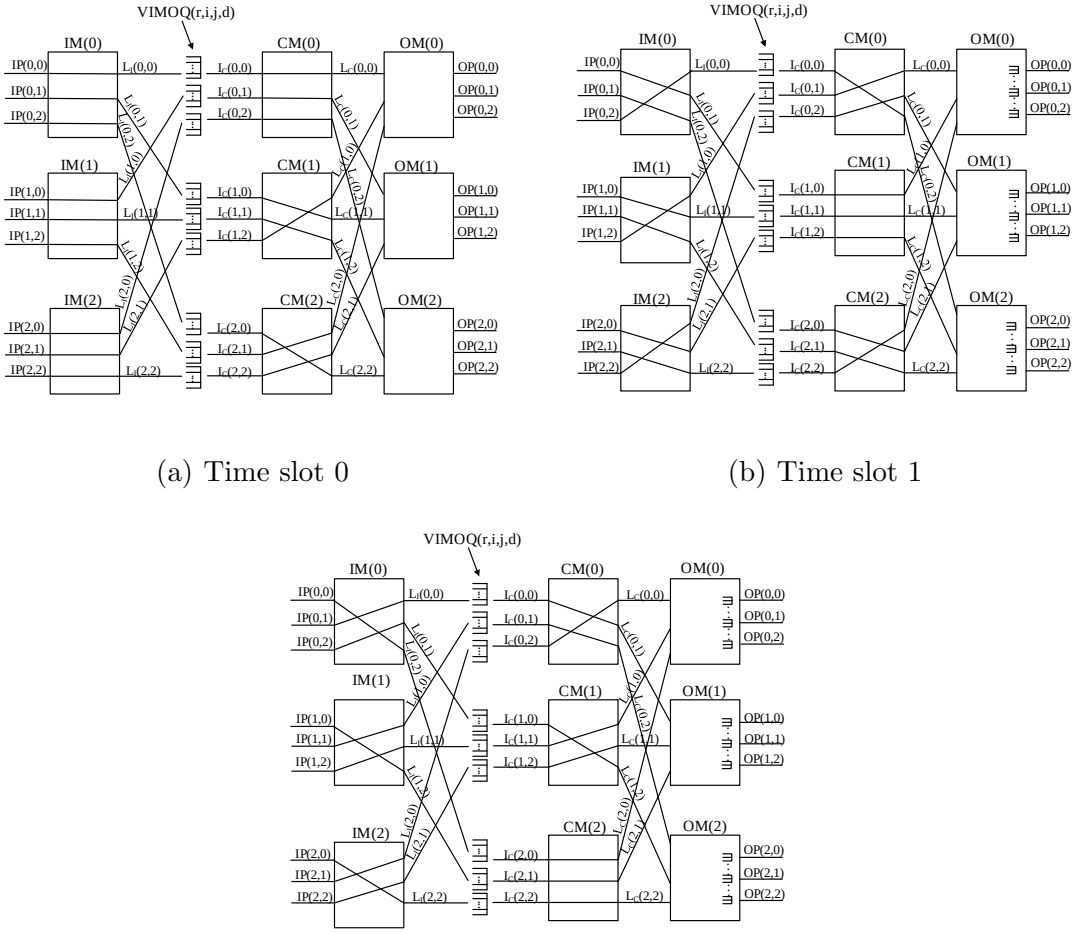
Table 3.2 shows an example of the configuration of the IMs and CMs of a 9×9 TRIDENT switch. Because $k = 3$, the example shows the configuration of three consecutive time slots. In this table, we use $w \rightarrow x$ to denote an interconnection between w and x. Figure 3.2 shows the configuration of the modules.

Table 3.2 Example of Configuration of Modules in a 9 × 9 TRIDENT Switch

Configuration						
Time slot	$IM(0)$	$CM(0)$	$IM(1)$	$CM(1)$	$IM(2)$	$CM(2)$
$t = 0$	$IP(0,0) \rightarrow L_I(0,0)$	$I_c(0,0) \rightarrow L_C(0,0)$	$IP(1,0) \rightarrow L_I(1,0)$	$I_c(1,0) \rightarrow L_C(1,1)$	$IP(2,0) \rightarrow L_I(2,0)$	$I_c(2,0) \rightarrow L_C(2,2)$
	$IP(0,1) \rightarrow L_I(0,1)$	$I_c(0,1) \rightarrow L_C(0,1)$	$IP(1,1) \rightarrow L_I(1,1)$	$I_c(1,1) \rightarrow L_C(1,2)$	$IP(2,1) \rightarrow L_I(2,1)$	$I_c(2,1) \rightarrow L_C(2,0)$
	$IP(0,2) \rightarrow L_I(0,2)$	$I_c(0,2) \rightarrow L_C(0,2)$	$IP(1,2) \rightarrow L_I(1,2)$	$I_c(1,2) \rightarrow L_C(1,0)$	$IP(2,2) \rightarrow L_I(2,2)$	$I_c(2,2) \rightarrow L_C(2,1)$
$t = 1$	$IP(0,0) \rightarrow L_I(0,1)$	$I_c(0,0) \rightarrow L_C(0,2)$	$IP(1,0) \rightarrow L_I(1,1)$	$I_c(1,0) \rightarrow L_C(1,0)$	$IP(2,0) \rightarrow L_I(2,1)$	$I_c(2,0) \rightarrow L_C(2,1)$
	$IP(0,1) \rightarrow L_I(0,2)$	$I_c(0,1) \rightarrow L_C(0,0)$	$IP(1,1) \rightarrow L_I(1,2)$	$I_c(1,1) \rightarrow L_C(1,1)$	$IP(2,1) \rightarrow L_I(2,2)$	$I_c(2,1) \rightarrow L_C(2,2)$
	$IP(0,2) \rightarrow L_I(0,0)$	$I_c(0,2) \rightarrow L_C(0,1)$	$IP(1,2) \rightarrow L_I(1,0)$	$I_c(1,2) \rightarrow L_C(1,2)$	$IP(2,2) \rightarrow L_I(2,0)$	$I_c(2,2) \rightarrow L_C(2,0)$
$t = 2$	$IP(0,0) \rightarrow L_I(0,2)$	$I_c(0,0) \rightarrow L_C(0,1)$	$IP(1,0) \rightarrow L_I(1,2)$	$I_c(1,0) \rightarrow L_C(1,2)$	$IP(2,0) \rightarrow L_I(2,2)$	$I_c(2,0) \rightarrow L_C(2,0)$
	$IP(0,1) \rightarrow L_I(0,0)$	$I_c(0,1) \rightarrow L_C(0,2)$	$IP(1,1) \rightarrow L_I(1,0)$	$I_c(1,1) \rightarrow L_C(1,0)$	$IP(2,1) \rightarrow L_I(2,0)$	$I_c(2,1) \rightarrow L_C(2,1)$
	$IP(0,2) \rightarrow L_I(0,1)$	$I_c(0,2) \rightarrow L_C(0,0)$	$IP(1,2) \rightarrow L_I(1,1)$	$I_c(1,2) \rightarrow L_C(1,1)$	$IP(2,2) \rightarrow L_I(2,1)$	$I_c(2,2) \rightarrow L_C(2,2)$

3.2.2 Arbitration at Output Ports

Each output port has a round-robin arbiter to keep track of the next flow to serve, and N flow pointers to keep track of the next cell to serve for each flow. As before, a flow is the set of cells from $IP(i, s)$ destined to $OP(j, d)$. An output port arbiter selects the flow to serve in a round-robin fashion. For this selection, the output arbiter selects the HoL cell of a CB if the cell's order matches the expected cell order for that flow. Because the output port arbiter selects the older cell based on the order of arrival to the switch, this selection prevents out-of-sequence forwarding. We discuss this property in Section 3.4. Furthermore, the round-robin schedule ensures fair service for different flows. If there is no HoL cell with the expected value for a particular flow, the arbiter moves to the next flow.



(a) Time slot 0

(b) Time slot 1

(c) Time slot 2

Figure 3.2 Configuration example of a 9 × 9 TRIDENT switch modules.

3.2.3 In-sequence Cell Forwarding Mechanism

The proposed in-sequence forwarding mechanism of TRIDENT is based on tagging cells of a flow at the inputs with their arriving sequence number, and forwarding cells from the crosspoint buffers to the output port in the same sequence they arrived in the input. The policy used for keeping cells in-sequence is as follows: When a cell of a flow arrives in the input port, the input port arbiter appends the arrival order to the cell (for the corresponding flow). After being forwarded through $L_I(i, r)$, the cell is stored at the VIMOQ for the destination OP. When the CM configuration permits, the cell is forwarded to the destined OM and stored at the queue for traffic from the IP to the destined OP traversing that CM. An OP arbiter selects cells of a flow in the order they arrived in the switch by using the arrival order carried by each cell. As an example of this operation, Table 3.3 shows the arrival times of cell $c_{1,1}$, $c_{2,1}$, and $c_{2,2}$, where c_{y,t_x} denotes flow y and arrival time t_x to the VIMOQs. Cell $c_{2,1}$ is queued behind $c_{1,1}$, and $c_{2,2}$ is placed in an empty VIMOQ. Table 3.4 shows the time slots when the cells are forwarded from the VIMOQ. For example, when $c_{2,2}$ leaves the VIMOQ before $c_{2,1}$. Table 3.5 shows the time slots when the cells are forwarded to the destination OP after the output-port arbitration is performed.

Table 3.3 Time Slots of Cell Arrival to VIMOQs in Example of the In-sequence Forwarding Mechanism

Cell arrival time		
t_x	t_{x+1}	t_{x+2}
$c_{1,1}$		
	$c_{2,1}$	$c_{2,2}$

Table 3.4 Time Slots of Cells Departure from VIMOQs in Example of the In-sequence Forwarding Mechanism

Cell departure time slots from VIMOQs						
t_x	t_{x+1}	t_{x+2}	t_{x+3}	t_{x+4}	t_{x+5}	t_{x+6}
			$c_{1,1}$			
				$c_{2,2}$		$c_{2,1}$

Table 3.5 Time Slots of Cells Departure from CBs in Example of the In-sequence Forwarding Mechanism.

Cell departure time slots from CBs								
t_x	t_{x+1}	t_{x+2}	t_{x+3}	t_{x+4}	t_{x+5}	t_{x+6}	t_{x+7}	t_{x+8}
				$c_{1,1}$				
							$c_{2,1}$	$c_{2,2}$

3.3 Throughput Analysis

In this section, we analyze the performance of the proposed TRIDENT switch. Figure 3.3 shows the block representation of TRIDENT and the traffic at each stage.

The block on the left indicates the IM stage, the block in the middle denotes the CM stage, and the small blocks on the right denote the OMs. Let us denote the traffic coming to the IMs, CMs, OMs, OPs, and the traffic leaving TRIDENT as $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, $\mathbf{R_4}$ and R_5, respectively. Here, $\mathbf{R_1}$ and $\mathbf{R_2}$, and $\mathbf{R_3}$ are $N \times N$ matrices, $\mathbf{R_4}$ comprises N $N \times 1$ column vectors, and $R_5(v)$ is a scalar.

The traffic from input ports to the IM stage, $\mathbf{R_1}$, is defined as:

$$\mathbf{R_1} = [\lambda_{u,v}] \tag{3.3}$$

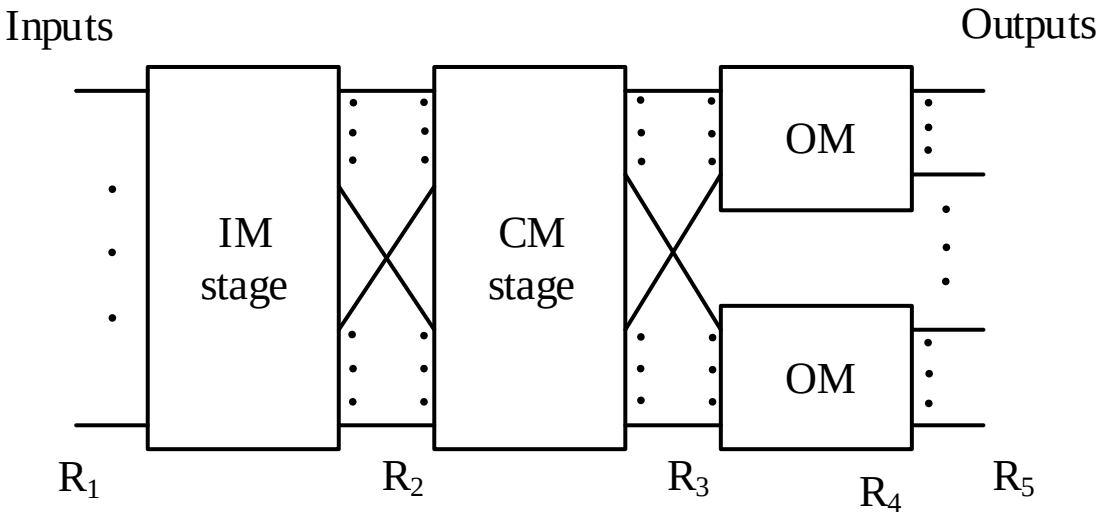


Figure 3.3 Block representation of the TRIDENT switch. The first block is the IM stage, the second block is the CM stage, and the last small blocks are the OMs.

Here, $\lambda_{u,v}$ is the arrival rate of traffic from input u to output v, where

$$u = ik + s \tag{3.4}$$

$$v = jm + d \tag{3.5}$$

and $0 \leq u, v \leq N - 1$.

In the following analysis, we consider admissible traffic, which is defined as:

$$\sum_{u=0}^{N-1} \lambda_{u,v} \leq 1, \ \sum_{v=0}^{N-1} \lambda_{u,v} \leq 1 \tag{3.6}$$

and as i.i.d. traffic.

The IM stage of TRIDENT balances the traffic load coming from the input ports to the VIMOQs. Specifically, the permutations used to configure the IMs forwards the traffic from an input to k different CMs, and then to the VIMOQs connected to these CMs in k consecutive time slots.

$\mathbf{R_2}$ is the traffic directed towards CMs and it is derived from $\mathbf{R_1}$ and the permutations of IMs. The configuration of the IM stage at time slot t that connects

$IP(i,s)$ to $L_I(i,r)$ are represented as an $N \times N$ permutation matrix, $\mathbf{\Pi}(t) = [\pi_{u,v}]$, where r is determined from (3.1) and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = rk + i \\ 0 & \text{elsewhere.} \end{cases}$$

The configuration of the IM stage can be represented as a compound permutation matrix, $\mathbf{P_1}$, which is the sum of the IM permutations over k time slots as follows,

$$\mathbf{P_1} = \sum^{k} \mathbf{\Pi}(t)$$

Because the configuration is repeated every k time slots, the traffic load from the same input going to each VIMOQ is $\frac{1}{k}$ of the traffic load of $\mathbf{R_1}$. Therefore, a row of $\mathbf{R_2}$ is the sum of the row elements of $\mathbf{R_1}$ at the non zero positions of $\mathbf{P_1}$, normalized by k. This is:

$$\mathbf{R_2} = \frac{1}{k}((\mathbf{R_1} * \mathbb{1}) \circ \mathbf{P_1}) \tag{3.7}$$

where $\mathbb{1}$ denotes an $N \times N$ unit matrix and $\circ$ denotes element/position wise multiplication, as before. There are k non-zero elements in each row of $\mathbf{R_2}$. Here, $\mathbf{R_2}$ is the aggregate traffic in all the VIMOQs destined to all OPs. This matrix can be further decomposed into k $N \times N$ submatrices, $\mathbf{R_2}(j)$, each of which is the aggregate traffic at VIMOQs designated for $OM(j)$.

$$\mathbf{R_2} = \sum_{j=0}^{j=k-1} \mathbf{R_2}(j) \tag{3.8}$$

where j is obtained from (3.5) $\forall\ d$. The configuration of the CM stage at time slot t that connects $I_c(r,p)$ to $L_{C(r,j)}$ may be represented as an $N \times N$ permutation matrix, $\mathbf{\Phi}(t) = [\phi_{u,v}]$, where j is determined from (3.2) and the matrix element:

$$\phi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = jk + r \\ 0 & \text{elsewhere.} \end{cases}$$

Similarly, the switching process at the CM stage is represented by a compound permutation matrix $\mathbf{P_2}$, which is the sum of k permutations of the CM stage over k

time slots. Here,

$$\mathbf{P_2} = \sum^{k} \mathbf{\Phi}(t)$$

The traffic forwarded to OMs, $\mathbf{R_3}(j)$, is:

$$\mathbf{R_3}(j) = \mathbf{R_2}(j) \circ \mathbf{P_2} \tag{3.9}$$

where j is obtained from (3.5) $\forall$ d. The traffic destined to $OP(j,d)$ at $OM(j)$, $\mathbf{R_3}(j,d)$, is obtained by extracting the traffic elements from $\mathbf{R_3}(j)$, or:

$$\mathbf{R_3}(j) = \sum_{d=0}^{d=k-1} \mathbf{R_3}(j,d) \tag{3.10}$$

where d is obtained from (3.5) for the different j. The aggregate traffic at CBs of an OP for the different IPs, $\mathbf{R_4}(v)$, is obtained from the multiplication of $\mathbf{R_3}(j,d)$ with a vector of all ones, $\vec{1}$, or:

$$\mathbf{R_4}(v) = \mathbf{R_3}(j,d) * \vec{1} \tag{3.11}$$

Each row of $\mathbf{R_4}(v)$ is the aggregate traffic at the CBs from each IP. The traffic leaving an OP, $R_5(v)$, is:

$$R_5(v) = (\vec{1})^T * \mathbf{R_4}(v) \tag{3.12}$$

Therefore, $\mathbf{R_5}(v)$ is the sum of the traffic leaving $OP(v)$.

The following example shows the operations performed on traffic coming to a 4×4 ($k = 2$) TRIDENT switch. Let the input traffic matrix be

$$\mathbf{R_1} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix}$$

First, $\mathbf{R_1}$ is decomposed into $\mathbf{R_2}$ at the IM stage. The compound permutation matrix for the IM stage for this switch is:

$$\mathbf{P_1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Using (3.7), we get

$$\mathbf{R_2} = 1/2 \begin{bmatrix} \sum_{i=0}^{3} \lambda_{0i} & 0 & \sum_{i=0}^{3} \lambda_{0i} & 0 \\ \sum_{i=0}^{3} \lambda_{1i} & 0 & \sum_{i=0}^{3} \lambda_{1i} & 0 \\ 0 & \sum_{i=0}^{3} \lambda_{2i} & 0 & \sum_{i=0}^{3} \lambda_{2i} \\ 0 & \sum_{i=0}^{3} \lambda_{3i} & 0 & \sum_{i=0}^{3} \lambda_{3i} \end{bmatrix}$$

From (3.8), the traffic matrix at VIMOQs destined for the different OMs are:

$$\mathbf{R_2}(0) = \frac{1}{2} \begin{bmatrix} \lambda_{0,0} + \lambda_{0,1} & 0 & \lambda_{0,0} + \lambda_{0,1} & 0 \\ \lambda_{1,0} + \lambda_{1,1} & 0 & \lambda_{1,0} + \lambda_{1,1} & 0 \\ 0 & \lambda_{2,0} + \lambda_{2,1} & 0 & \lambda_{2,0} + \lambda_{2,1} \\ 0 & \lambda_{3,0} + \lambda_{3,1} & 0 & \lambda_{3,0} + \lambda_{3,1} \end{bmatrix}$$

$$\mathbf{R_2}(1) = \frac{1}{2} \begin{bmatrix} \lambda_{0,2} + \lambda_{0,3} & 0 & \lambda_{0,2} + \lambda_{0,3} & 0 \\ \lambda_{1,2} + \lambda_{1,3} & 0 & \lambda_{1,2} + \lambda_{1,3} & 0 \\ 0 & \lambda_{2,2} + \lambda_{2,3} & 0 & \lambda_{2,2} + \lambda_{2,3} \\ 0 & \lambda_{3,2} + \lambda_{3,3} & 0 & \lambda_{3,2} + \lambda_{3,3} \end{bmatrix}$$

The rows of $\mathbf{R_2}(v)$ represent the traffic from IPs, and the columns represent $VIMOQ(r, i, j, d)$ at $I_C(r, p)$. The compound permutation matrix for the CM stage

for this switch is:

$$\mathbf{P_2} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

From (3.9) and (3.10), the traffic forwarded to an OP is:

$$\mathbf{R_3}(0,0) = \tfrac{1}{2}\begin{bmatrix} \lambda_{0,0} & 0 & \lambda_{0,0} & 0 \\ \lambda_{1,0} & 0 & \lambda_{1,0} & 0 \\ 0 & \lambda_{2,0} & 0 & \lambda_{2,0} \\ 0 & \lambda_{3,0} & 0 & \lambda_{3,0} \end{bmatrix} \qquad \mathbf{R_3}(0,1) = \tfrac{1}{2}\begin{bmatrix} \lambda_{0,1} & 0 & \lambda_{0,1} & 0 \\ \lambda_{1,1} & 0 & \lambda_{1,1} & 0 \\ 0 & \lambda_{2,1} & \lambda_{2,1} & \\ 0 & \lambda_{3,1} & 0 & \lambda_{3,1} \end{bmatrix}$$

$$\mathbf{R_3}(1,0) = \tfrac{1}{2}\begin{bmatrix} \lambda_{0,2} & 0 & \lambda_{0,2} & 0 \\ \lambda_{1,2} & 0 & \lambda_{1,2} & 0 \\ 0 & \lambda_{2,2} & 0 & \lambda_{2,2} \\ 0 & \lambda_{3,2} & 0 & \lambda_{3,2} \end{bmatrix} \qquad \mathbf{R_3}(1,1) = \tfrac{1}{2}\begin{bmatrix} \lambda_{0,3} & 0 & \lambda_{0,3} & 0 \\ \lambda_{1,3} & 0 & \lambda_{1,3} & 0 \\ 0 & \lambda_{2,3} & 0 & \lambda_{2,3} \\ 0 & \lambda_{3,3} & 0 & \lambda_{3,3} \end{bmatrix}$$

The rows of $\mathbf{R_3}(j,d)$ represent the traffic from $VIMOQ(r,i,j,d)$ at $I_C(r,p)$ and the columns represent $L_C(r,j)$.

The traffic forwarded from CBs allocated for the different IPs to the corresponding OP is obtained from (3.11):

$$\mathbf{R_4}(0) = \begin{bmatrix} \lambda_{0,0} \\ \lambda_{1,0} \\ \lambda_{2,0} \\ \lambda_{3,0} \end{bmatrix}, \qquad \mathbf{R_4}(1) = \begin{bmatrix} \lambda_{0,1} \\ \lambda_{1,1} \\ \lambda_{2,1} \\ \lambda_{3,1} \end{bmatrix}, \qquad \mathbf{R_4}(2) = \begin{bmatrix} \lambda_{0,2} \\ \lambda_{1,2} \\ \lambda_{2,2} \\ \lambda_{3,2} \end{bmatrix}, \qquad \mathbf{R_4}(3) = \begin{bmatrix} \lambda_{0,3} \\ \lambda_{1,3} \\ \lambda_{2,3} \\ \lambda_{3,3} \end{bmatrix}$$

The rows of $\mathbf{R_4}(v)$ represent the traffic from $IP(i,s)$. Using (3.12), we obtain the sum of the traffic leaving the OP, or:

$R_5(0) = \sum_{i=0}^{3} \lambda_{i0}, \qquad R_5(1) = \sum_{i=0}^{3} \lambda_{i1}, \qquad R_5(2) = \sum_{i=0}^{3} \lambda_{i2}, \qquad R_5(3) = \sum_{i=0}^{3} \lambda_{i3}$

Theorem 3. *The TRIDENT achieves 100% throughput under any admissible i.i.d traffic.*

Proof: From $\mathbf{R_4}(0)$ to $\mathbf{R_4}(3)$ above, we can deduce that $\mathbf{R_4}$ is equal to the input traffic $\mathbf{R_1}$, or:

$$\mathbf{R_4}(v) = \mathbf{R_1}(v) \; \forall \; v \tag{3.13}$$

Also, since $\mathbf{R_2}$ and $\mathbf{R_4}(v)$ meet the admissibility condition in (3.6), and $R_5(v)$ does not exceed the service rate for any $OP(v)$, this implies that the sum of the traffic load at each $VIMOQ$, CB, and OP does not exceed their respective capacities. From the admissibility of $\mathbf{R_2}$ and $\mathbf{R_4}(v)$, and (3.13), we can infer that the input traffic are successfully forwarded to the output ports.

As discussed in Section 3.2.2, the output arbiter selects a flow in a round-robin fashion and a cell within that flow based on it's order of arrival. If a cell of a flow is not selected, the OP arbiter moves to the next flow. This ensures fairness and that the cells forwarded to the OP are also forwarded out of the OP. Hence, from $R_5(0)$ to $R_5(3)$ above, we can infer that $R_5(v)$ is equal to $\mathbf{R_4}(v)$, or:

$$R_5(v) = (\vec{1})^T * \mathbf{R_4}(v) \; \forall \; v \tag{3.14}$$

From (3.13) and (3.14), we can conclude that TRIDENT can achieve 100% throughput under admissible traffic.

This completes the proof of Theorem 3. ■

3.4 Analysis of In-Sequence Service

In this section, we demonstrate that the TRIDENT switch forwards cells in sequence to the OPs through the proposed in-sequence forwarding mechanism. Table 3.6 lists the terms used in the in-sequence analysis of the proposed TRIDENT switch. Here, $c_{y,\tau}(i, s, j, d)$ denotes the τth cell of traffic flow y, which comprises cells going from

$IP(i,s)$ to $OP(j,d)$. In addition, $t_{a_{y,\tau}}$ denotes the arrival time of $c_{y,\tau}$, and $q_{V_{y,\tau}}$ and $q_{C_{y,\tau}}$ denote the queuing delays experienced by $c_{y,\tau}$ at $VIMOQ(r,i,j,d)$ and $CB(r,j,d,i,s)$, respectively. The departure times of $c_{y,\tau}$ from the corresponding VIMOQ and CB are denoted as $d_{V_{y,\tau}}$ and $d_{C_{y,\tau}}$, respectively. We consider admissible traffic in this analysis.

Here, we claim that TRIDENT forwards cells in sequence to the output ports, through the following theorem.

Theorem 4. *For any two cells $c_{y,\tau}(i,s,j,d)$ and $c_{y,\tau'}(i,s,j,d)$, where $\tau < \tau'$, $c_{y,\tau}(i,s,j,d)$ departs the destined output port before $c_{y,\tau'}(i,s,j,d)$.*

Table 3.6 Notations used in the In-sequence Analysis of TRIDENT

$c_{y,\tau}$	The τth cell of flow y from $IP(i,s)$ to $OP(j,d)$.
$t_{a_{y,\tau}}$	Arrival time of $c_{y,\tau}$ at $IP(i,s)$.
$N_{V_{y,\tau}}$	The number of cells at $VIMOQ(r,i,j,d)$ upon the arrival of $c_{y,\tau}$.
$q_{H_{y,\tau}}$	The residual queuing delay of the HoL cell at $VIMOQ(r,i,j,d)$ upon the arrival of $c_{y,\tau}$.
$q_{V_{y,\tau}}$	Queuing delay of $c_{y,\tau}$ at $VIMOQ(r,i,j,d)$.
$d_{V_{y,\tau}}$	Departure time of $c_{y,\tau}$ from $VIMOQ(r,i,j,d)$ at $I_C(r,p)$.
$N_{C_{y,\tau}}$	The number of cells at $CB(r,j,d,i,s)$ upon the arrival of $c_{y,\tau}$.
$q_{C_{y,\tau}}$	Queuing delay of $c_{y,\tau}$ at $CB(r,j,d,i,s)$ of $OP(j,d)$.
$d_{C_{y,\tau}}$	Departure time of $c_{y,\tau}$ from $CB(r,j,d,i,s)$.

Lemma 4. *For any flow traversing the TRIDENT switch, an older cell is placed ahead of a younger cell from the same flow in the same crosspoint buffer.*

Proof: From the architecture and configuration of the switch an IP connects to a CM once every k time slots. If a younger cell arrives at the OM before an older cell then the younger cell was forwarded through a different CM from the one the older cell was buffered. Also, two cells of the same flow may be queued in the same CB if

and only if the younger cell arrived at the VIMOQ k time slots later than the older cell, and therefore, the younger cell would be lined up in a queue position behind the position of the older cell.

■

Lemma 5. *For any number of flows traversing the TRIDENT switch, cells from the same flow are cleared from the OP in the same order they arrived at the IP.*

Proof: Let us consider a traffic scenario where there are multiple flows traversing the switch. We focus on one flow with cells arriving back to back. Let us also consider as an initial condition that all CBs are empty, and that the VIMOQ to where the first cell of the flow is sent has backlogged cells (from other flows) while other VIMOQs to where the subsequent cells of the same flow are sent are empty. This scenario would have the largest probability to delay the first cell of the flow and, therefore; to forward the subsequent cells of the flow out of sequence. Also, let us consider that the flow pointer at the output ports initially points to the cell arrival order $L_{y\theta}$, where y is the flow id and θ is the cell's order of arrival.

Also, let us assume that the cells arrive at $L_I(i, r)$ one or more time slots before the configuration of the CM allows forwarding a cell to its destined OM. Thus, a cell may depart in the following or a few time slots after its arrival. This cell then may wait up to $k - 1$ time slots for a favorable interconnection to take place at the CM before being forwarded to the destined OM. In the remainder of the discussion, we show that the arriving cells are forwarded to the destination OP in the same order they arrive in the IP.

Given flow y, the arrival time of the first cell $c_{y,\tau}$ is:

$$t_{a_{y,\tau}} = t_x \tag{3.15}$$

Upon arriving in the IP, $c_{y,\tau}$ is tagged with L_{y0} and forwarded to the VIMOQ. Based on the backlog condition, $c_{y,\tau}$ is placed behind γ cells from other flows upon arriving at the VIMOQ. Therefore, the VIMOQ occupancy, $N_{V_{y,\tau}}$, is:

$$N_{V_{y,\tau}} = \gamma \tag{3.16}$$

Using (3.16) the queuing delay of $c_{y,\tau}$ at the VIMOQ is:

$$q_{V_{y,\tau}} = q_{H_{y,\tau}} + (\gamma - 1)k + k \tag{3.17}$$

where $q_{H_{y,\tau}}$ is the time it takes the HoL cell to depart the VIMOQ and $(\gamma - 1)k$ is the delay generated by the other $(\gamma - 1)$ cells ahead of $c_{y,\tau}$ in the VIMOQ. The extra k time slots are the delay $c_{y,\tau}$ experiences as it waits for the configuration pattern to repeat after the last cell ahead of it is forwarded to the OM.
Using (3.15) and (3.17), the departure time of $c_{y,\tau}$ from the VIMOQ is:

$$d_{V_{y,\tau}} = t_{a_{y,\tau}} + q_{H_{y,\tau}} + \gamma k \tag{3.18}$$

When $c_{y,\tau}$ arrives at the output module it is stored at the corresponding output buffer before being forwarded to the output port.

Let us now consider the next arriving cell from flow y, $c_{y,\tau+\theta}$, where $0 < \theta < k$. The time of arrival of $c_{y,\tau+\theta}$ is:

$$t_{a_{y,\tau+\theta}} = t_x + \theta \tag{3.19}$$

Upon arrival, $c_{y,\tau+\theta}$ would have $L_{y\theta}$ appended to it and forwarded to the VIMOQ. Based on the traffic scenario, $c_{y,\tau+\theta}$ would be forwarded to an empty VIMOQ. The queuing delay at the $VIMOQ$ for $c_{y,\tau+\theta}$ is:

$$q_{V_{y,\tau+\theta}} = \beta \tag{3.20}$$

where β is the number of time slots before the configuration pattern enables forwarding $c_{y,\tau+\theta}$ to the destined OM. Using (3.18), (3.19), and (3.20), the departure time of $c_{y,\tau+\theta}$ from the VIMOQ is:

$$d_{V_{y,\tau+\theta}} = t_x + \theta + \beta \tag{3.21}$$

At the output port, the pointers all initially pointed to L_{y0} based on the initial condition. Therefore, irrespective of $d_{V_{y,\tau+\theta}} < d_{V_{y,\tau}}$, for $\theta + \beta < q_{H_{y,\tau}} + \gamma k$, $c_{y,\tau+\theta}$ remains stored at the output buffer until $c_{y,\tau}$ is cleared from the output port, because the pointer points to L_{y0}. Because CBs are empty as initial condition, the CB occupancy, $N_{C_{y,\tau}}$, upon $c_{y,\tau}$ arrival is:

$$N_{C_{y,\tau}} = 0 \tag{3.22}$$

and the occupancy of the CB, $N_{C_{y,\tau+\theta}}$, upon $c_{y,\tau+\theta}$ arrival is

$$N_{C_{y,\tau+\theta}} = 0 \tag{3.23}$$

Using (3.22), the queuing delay, $q_{C_{y,\tau}}$, at the CB for $c_{y,\tau}$ is:

$$q_{C_{y,\tau}} = 0 \tag{3.24}$$

From (3.18), (3.21), and (3.23), the queuing delay, $q_{C_{y,\tau+\theta}}$, at the CB for $c_{y,\tau+\theta}$ is:

$$q_{C_{y,\tau+\theta}} = q_{H_{y,\tau}} + \gamma k - \beta \tag{3.25}$$

From (3.15), (3.18), and (3.24), the departure time of $c_{y,\tau}$ from the OP, $d_{C_{y,\tau}}$, is:

$$d_{C_{y,\tau}} = t_x + 1 + q_{H_{y,\tau}} + \gamma k \tag{3.26}$$

From (3.19), (3.21), and (3.25), the departure time of $c_{y,\tau+\theta}$ from the OP, $d_{C_{y,\tau+\theta}}$, is:

$$d_{C_{y,\tau+\theta}} = t_x + 1 + \theta + q_{H_{y,\tau}} + \gamma k \tag{3.27}$$

Using (3.26) and (3.27),

$$d_{C_{y,\tau+\theta}} - d_{C_{y,\tau}} = \theta \tag{3.28}$$

The difference between the departure times of any two cells of a flow from the CB is a function of θ, which is the arrival time difference between any two cells. Therefore, cells of a flow are forwarded to the OP in the same order they arrived.

■

This completes the proof of Theorem 4.

■

3.5 Performance Analysis

We evaluated the performance of TRIDENT through computer simulation under uniform traffic model and compared with that of an output-queued (OQ) and a Memory-Memory-Memory Clos-network (MMM) switch. We also evaluated the performance of TRIDENT through computer simulation under nonuniform traffic model and compared with that of an output-queued (OQ), Memory-Memory-Memory Clos-network (MMM), and MMM switch with extended memory (MM^{e}M) switches. The MMM switch selects cells from the buffers in the previous stage modules using forwarding arbitration schemes and is prone to serving cells out of sequence. Considering that most load-balancing switches based on Clos networks deliver low performance, we select these switches for comparison because they achieve the highest performance among Clos-network switches, despite been categorized as different architectures. We considered switches with size $N = \{64, 256\}$. For performance analysis, queues are assumed long to avoid cell losses and to identify average cell delay.

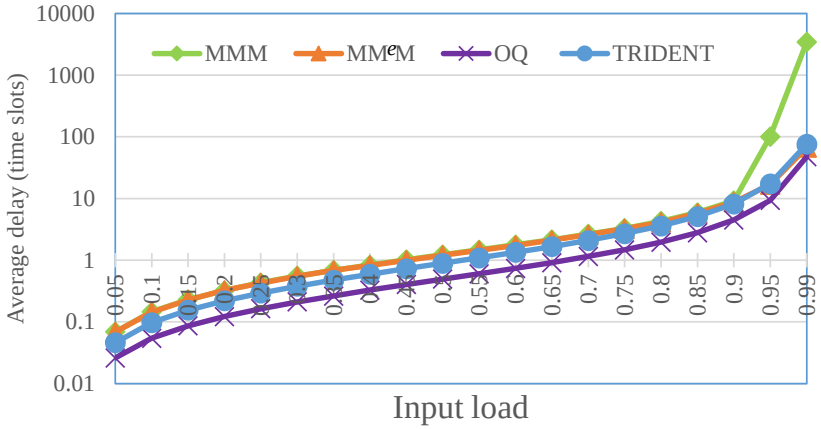


Figure 3.4 Average queueing delay under uniform traffic for N=64.

3.5.1 Uniform Traffic

Uniform distribution is mostly considered to be benign and the average rate for each output port $\lambda_{i,s,j,d} = \frac{1}{N}$. where $IP(i,s)$ is the source IP and $OP(j,d)$ is the destination OP. Hence, a packet arriving at the IP has an equal probability to be destined to any OP. Figures 3.4 and 3.5 show the average under uniform traffic with Bernoulli arrivals for $N = 64$ and $N = 256$, respectively. The finite and moderate average queuing delay indicated by the results shows that TRIDENT achieves 100% throughput under this traffic pattern. This is the result of the efficient load-balancing process in the IM stage. However, such a high performance is expected for uniformly distributed input traffic.

TRIDENT switch experiences a slightly higher average delay than the OQ switch, this is a result of cells being queued in the VIMOQs until a configuration occurs that enables forwarding the cells to their destined output modules. Due to the amount of memory required by MM^{e}M to implement the extended set of queues, our simulator can only simulate small MM^{e}M switches for queueing analysis, so we simulated the switches under this traffic pattern for $N = 64$. This figure also shows that TRIDENT achieves a lower average delay than the MMM switch.

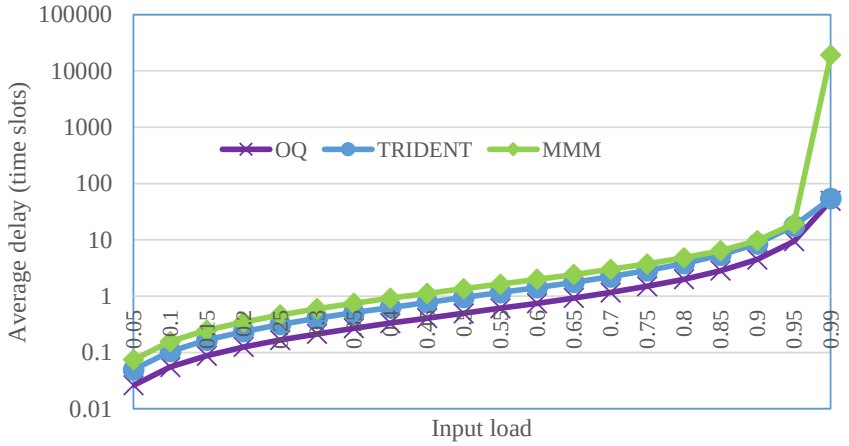


Figure 3.5 Average queueing delay under uniform traffic for N=256.

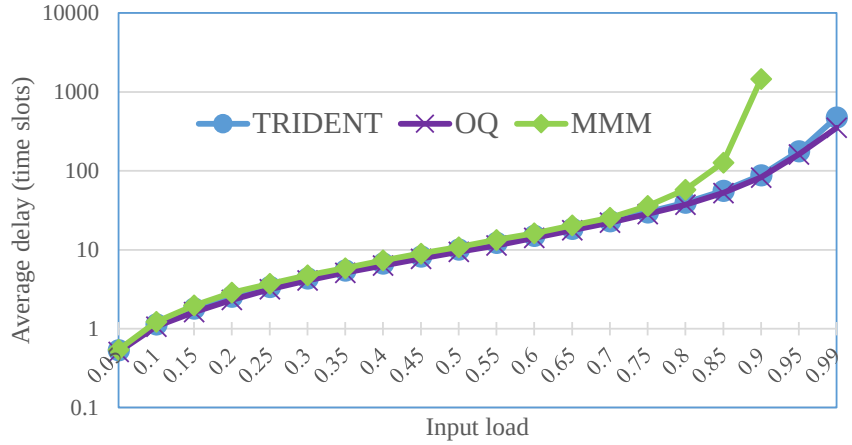


Figure 3.6 Average queuing delay under uniform bursty traffic with average burst length l=10.

Uniform bursty traffic is modeled as an ON-OFF Markov modulated process, with an average duration of the ON period set as the average burst length, l, with $l = \{10, 30\}$ cells. Figures 3.6 and 3.7 show the average delay under uniform traffic with bursty arrivals for average burst length of 10 and 30 cells, respectively. The results show that TRIDENT achieves 100% throughput under bursty uniform traffic and it is not affected by the burst length, while the MMM switch has a throughput of 0.8 and 0.75 for an average burst length of 10 and 30 cells, respectively. Therefore, TRIDENT achieves a performance closer to that of the OQ switch.

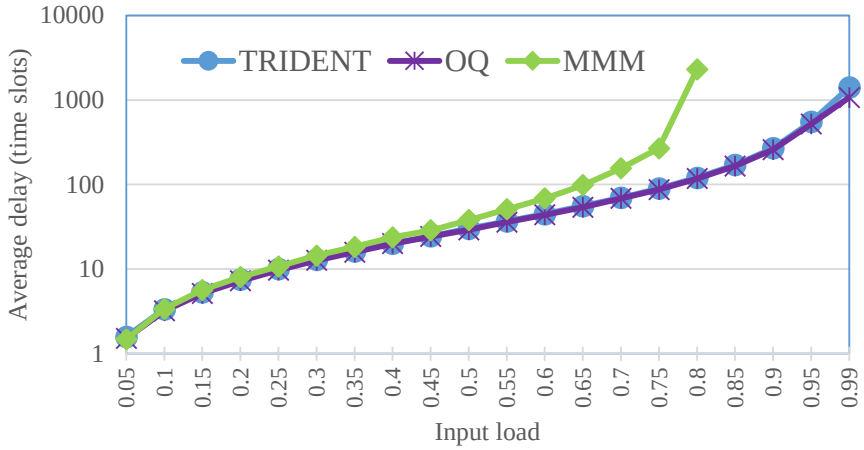


Figure 3.7 Average queuing delay under uniform bursty traffic with average burst length l=30.

The uniform distribution of the traffic and the load-balancing stage helps to attain this low queueing delay and high throughput. Figures 3.4, 3.5, 3.6, and 3.7 show that the queueing delay difference between TRIDENT and the OQ switch is not significant. These figures also show that the effective load balancing reduces the average delay and also eliminates the offset in delay for light load.

3.5.2 Nonuniform Traffic

We also evaluated the performance of TRIDENT, MMM, MM^{e}M, and OQ switches under nonuniform traffic. We adopted the unbalanced traffic model [72, 74] as a nonuniform traffic pattern. The nonuniform traffic can be modeled using an unbalanced probability ω to indicate the load variances for different flows. Consider input port $IP(i, s)$ and output port $OP(j, d)$ of the TRIDENT switch, the traffic load is determined by

$$\rho_{i,s,j,d} = \begin{cases} \rho(\omega + \dfrac{1-\omega}{N}), & \text{if } i = j \text{ and } s = d, \\ \rho\dfrac{1-\omega}{N}, & \text{otherwise} \end{cases} \tag{3.29}$$

where ρ is the input load for input $IP(i, s)$ and ω is the unbalanced probability. When ω=0, the input traffic is uniformly distributed and when ω=1, the input traffic is completely directional; traffic from $IP(i, s)$ is destined for $OP(j, d)$.

Figure 3.8 shows the throughput of TRIDENT, MMM, and MM^{e}M switches. The figure shows that TRIDENT switch attains 100% throughput under this traffic pattern for all values of ω, matching the performance of MM^{e}M and outperforming that of MMM. These two buffered switches are known to achieve high throughput at the expense of out-of-sequence forwarding.

We also tested the average queueing delay of TRIDENT under this nonuniform traffic. It has been shown that many switches do not achieve high throughput when ω is around 0.6 [72]. Therefore, we measured the average delay of TRIDENT under this unbalanced probability, as Figures 3.9 and 3.10 show for $N = 64$ and $N = 256$, respectively, and compared it with MMM, MM^{e}M, and OQ switches. It should be noted that due to the limited scalability of MMM and MM^{e}M, the comparison of TRIDENT for $N = 256$ under this traffic conditions only includes an OQ switch.

As Figure 3.9 for $N = 64$ shows, the average delay of TRIDENT is lower than the delay achieved by MMM and MM^{e}M under high input loads while also achieving a comparable delay of an OQ switch. The small performance difference between TRIDENT and OQ is similar for $N = 256$, as Figure 3.10 shows. These results are achieved because the load-balancing stage of TRIDENT distributes the traffic uniformly throughout the switch. Therefore, the queuing delay is similar to that observed under uniform traffic. These results also show that high switching performance of TRIDENT is not affected by the in-sequence mechanism of the switch and the load-balancing effect is more noticeable under nonuniform traffic.

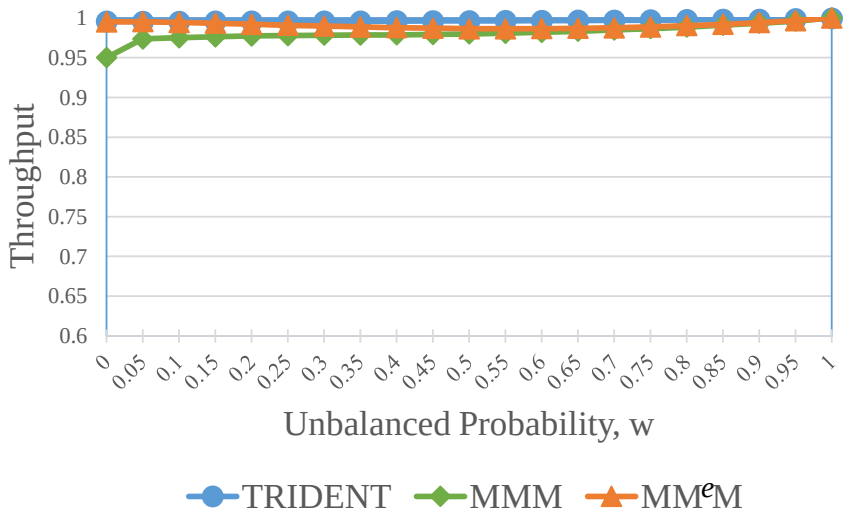


Figure 3.8 Throughput under unbalanced traffic for $0 \leq w \leq 1.0$ and N=256.

3.6 Conclusions

We have introduced a three-stage load-balancing packet switch that has virtual output module queues between the input and central stages, and a low-complexity scheme for configuration and forwarding cells in sequence for this switch. We call this switch TRIDENT. To effectively perform load balancing TRIDENT has virtual output module queues between the IM and CM stages. Here, IMs and CMs are bufferless modules, while the OMs are buffered ones. All the bufferless modules follow a predeterministic configuration while the OM selects the cell of a flow to be forwarded to destined output port based on the cell's arrival order and uses round-robin scheduling to select the flow to be served. Therefore, the switch does not have to pursue port matching despite having bufferless modules, and the configuration complexity of the switch is minimum, making it comparable to that of MMM switches. We introduced an in-sequence mechanism that operates at the outputs based on arrival order inserted at the inputs of TRIDENT to avoid out-of-sequence forwarding caused by the central buffers. We modeled and analyzed the operations that each of the stages effects on the incoming traffic to obtain the loads seen by the output ports. We showed that for admissible independent and identically distributed traffic,

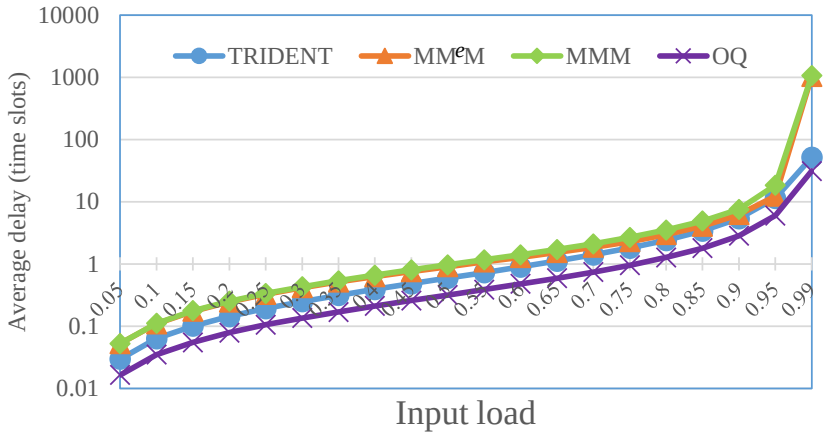


Figure 3.9 Average queuing delay under unbalanced traffic with $w = 0.6$ for N=64.

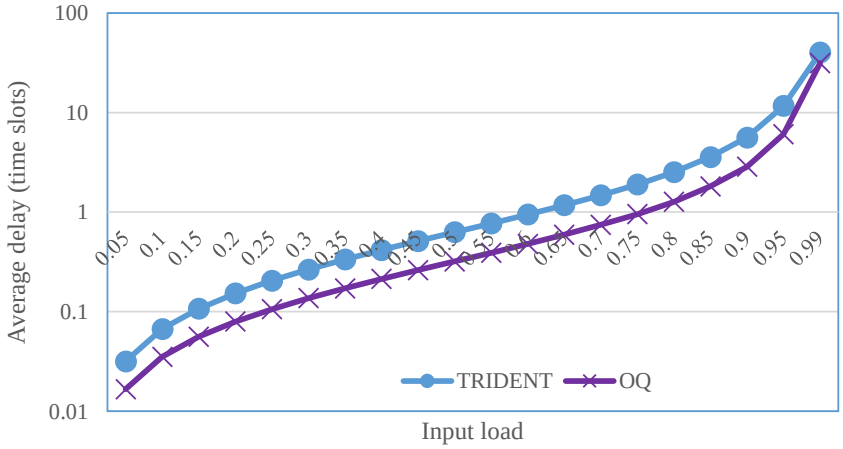


Figure 3.10 Average queuing delay under unbalanced traffic with $w = 0.6$ for N=256.

the switch achieves 100% throughput. This high performance is achieved without resorting to speedup nor switch expansion. In addition, we analyzed the operation of the forwarding mechanism and demonstrated that it forwards cells in sequence. We showed, through computer simulation, that for all tested traffic, the switch achieves 100% throughput for uniform and nonuniform traffic distributions.

CHAPTER 4

VINE: LOAD-BALANCING CLOS-NETWORK SWITCH WITH VIRTUAL INPUT-MODULE OUTPUT QUEUES AT CENTRAL STAGE

4.1 Introduction

Although Clos-network switches reduce the amount of hardware required for their implementation, configuration complexity stills limits their scalability [23]. In chapters 2 and 3, we introduced LBC and TRIDENT, respectively. But LBC requires an extra stage and its delay is high as compared to an OQ switch, while TRIDENT requires too many queues per output port for its in-sequence mechanism and also its delay is high when compared to an OQ switch. These factors might limit the scalability of a switch. In this chapter, we address these limiting factors by proposing a novel load-balancing Clos-network switch with Virtual Input module at ceNtral stagE (VINE). The proposed switch has virtual output queues (VOQs) at input ports, virtual input-module output queues (VIMOQs) at CMs, and crosspoint buffers (CBs) at OMs. The switch is based on cell switching, this is, a variable-length packet is segmented into cells at arrival, and it is reassembled before departure at output ports. The transmission time of a cell defines a time slot. We also propose using periodic configurations at IMs to load-balance traffic with minimum IM configuration complexity. CMs and OMs are buffered crossbars and crosspoint buffers at CMs are split to accommodate one queue per IM and per output. Incoming traffic is then load-balanced at IMs and forwarded to VIMOQs at CMs, then cells are forwarded to CBs at OMs, and finally sent through output ports. We also propose an in-sequence mechanism, operating at the input ports, to forward cells in sequence as queueing at CMs and OMs raise the possibility of out-of-sequence forwarding. The configuration

of VINE is low as no matching is required; IMs follow deterministic and periodic configurations, and CMs and OMs use simple arbitration to select a queue. The application of traffic load-balancing at IMs and buffering at CMs and OMs enable VINE to attain 100% throughput under admissible traffic with independent and identical distributions (i.i.d.) while resorting to low-complexity configuration and using an in-sequence mechanism. Our contribution is the proposal of VINE, as a high-performance and scalable Clos-network packet switch. The performance of VINE is comparable to that of an output queue (OQ) switch in terms of throughput and average delay on the tested traffic patterns. This high performance and in-sequence forwarding of VINE are both achieved without memory speedup nor central-stage expansion.

The remainder of this chapter is organized as follows: In Section 4.2, we introduce the VINE switch. In Section 4.3, we analyze the throughput performance of the proposed switch and proves that it attains 100% throughput. In Section 4.4, we analyze the delay of the proposed switch. In Section 4.5, we present a simulation study on the performance of the proposed switch. In Section 4.6, we present our conclusions.

4.2 Switch Architecture

VINE has N input ports (IPs) and N output ports (OPs), each denoted as $IP(i, s)$ and $OP(j, d)$, respectively, where $0 \leq i,\ j \leq k-1$, $0 \leq s,\ d \leq n-1$, and $N = nk$. In addition, $n = k = m$ for symmetry and cost effectiveness. The switch has k $n \times k$ IMs, m $k \times k$ CMs, and k $k \times n$ OMs. Figure 4.1 shows the architecture of VINE. Table 4.1 lists the notations used in the description of VINE. A VOQ, $VOQ(i, s, j, d)$, stores cells at $IP(i, s)$ for $OP(j, d)$. An IM is a bufferless crossbar and an OM and CM are buffered ones. In addition, a crosspoint buffer (queue) of CM is split per OP. Each VIMOQ is denoted as $VIMOQ(i, r, j, d)$, where $0 \leq r \leq m-1$.

Each IM output, $L_I(i,r)$, connects to $CM(r)$. The k VIMOQs at a CM directed to $OM(j)$, or $\sum_{0 \le d \le k-1} VIOMQ(i,r,j,d)$, contend for access to the CM output link, $L_C(r,j)$. Each OP has k crosspoint buffers, each denoted as $CB(j,d,r)$ and designated for queueing cells from $CM(r)$. VINE uses a flow control mechanism to avoid queue overflow and underflow between VIMOQs and VOQs, and between CBs and VIMOQs. The flow control mechanism uses counters to keep track of occupancy and two thresholds to stop and continue forwarding. The settings of thresholds considers propagation delays between queues, in number of cells. Therefore, a CB may be large enough to store multiple cells. Cells sent from IPs are load-balanced to VIMOQs and queued before they are forwarded to their destination OPs.

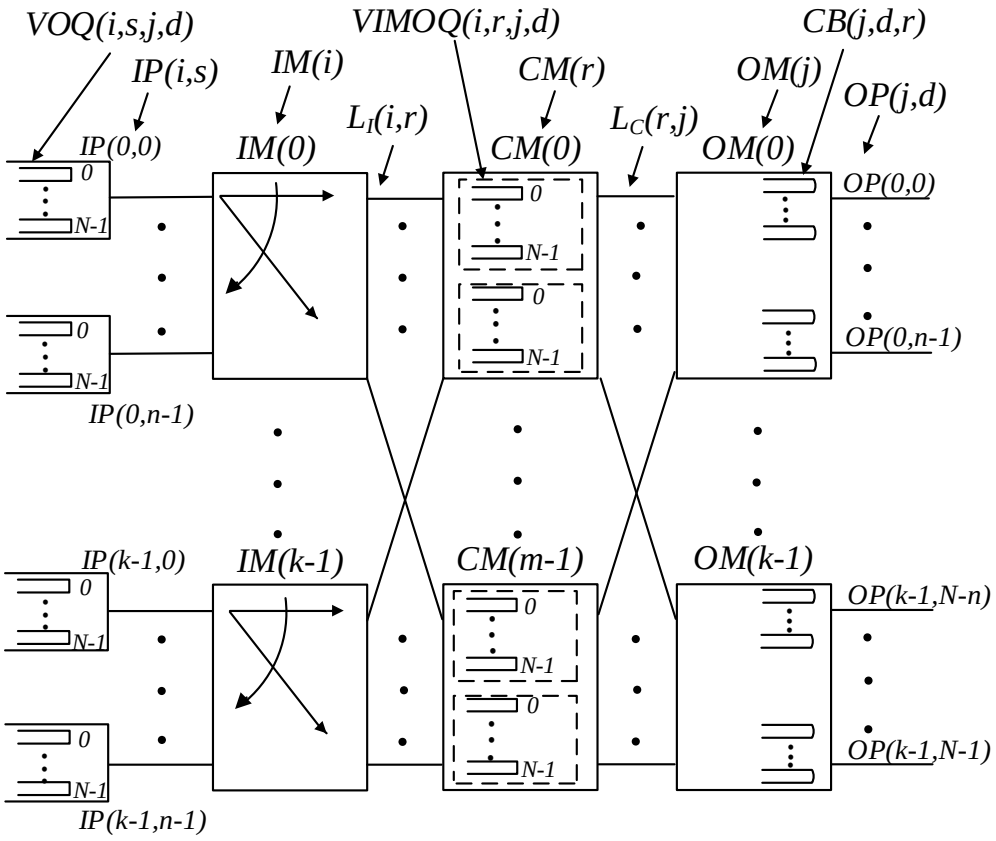


Figure 4.1 VINE: load-balancing Clos-network Switch with Virtual Input-module output queues at CMs (VINE) and crosspoint buffers at OMs.

Table 4.1 Notations used in the Description of the VINE Switch

Term	Description
N	Number of input/output ports.
n	Number of input/output ports for each IM and OM.
k	Number of IMs and OMs, where $k = \frac{N}{n}$.
$IP(i, s)$	Input port s of $IM(i)$, where $0 \leq i \leq k-1, 0 \leq s \leq n-1$.
$IM(i)$	Input module i.
$CM(r)$	Central module r.
$OM(j)$	Output module j, where $0 \leq j \leq k-1$.
$VOQ(i, s, j, d)$	VOQ at $IP(i, s)$ that stores cells destined to $OP(j, d)$, where $0 \leq d \leq n-1$.
$VIMOQ(i, r, j, d)$	VIMOQ at output of IMs that stores cells destined to $OP(j, d)$.
$L_I(i, r)$	Output link of $IM(i)$ connected to $CM(r)$.
$L_C(r, j)$	Output link of $CM(r)$ connected to $OM(j)$.
$CB(j, d, r)$	Crosspoint buffer at $OM(j)$ that stores cells going through $CM(r)$ and destined to $OP(j, d)$.
$OP(j, d)$	Output port d at $OM(j)$.

4.2.1 Input Module Configuration

Each IP uses a round-robin policy to select a VOQ from which a cell is forwarded to a CM. Eligible VOQs have one or more cells and are enabled by the flow control mechanism.

Each IM uses a fixed and pre-determined sequence of k permutations, one for each time slot, at IMs to reduce configuration complexity. Each permutation consists of a unique input-output interconnection and such that any IP is connected to any other OP after k permutations. Cells at IPs are forwarded to the output of the IMs based on the configuration at that time slot and stored in the VIMOQ corresponding to its destination OP.

At time slot t, $IM(i)$ is configured to interconnect input $IP(i, s)$ to $L_I(i, r)$, where

$$r = (s + t) \mod k \tag{4.1}$$

4.2.2 Selection of VIMOQ at Central Modules.

Each $L_C(r, j)$ uses a hierarchical static round-robin arbiter that selects a cell to be forwarded to the OM, and thus to the corresponding CB at the destination OP. The arbiter selects the IM first and the destined OP second. Each $L_C(r, j)$ also has pointers to keep track of the selected IM and OP to avoid starvation.

4.2.3 Selection at Output Ports.

VINE uses round-robin arbitration at an OP to select the CB that forwards a cell to the OP in the following time slot. Because there is no two (or more) cells of the same flow at CBs, as per the in-sequence mechanism applied to IPs, round-robin arbitration at OPs provides service to CBs without concerns of out-of-sequence forwarding. As before, a flow is the set of cells from $IP(i, s)$ destined to $OP(j, d)$.

4.2.4 In-sequence Cell Forwarding Mechanism

Cells of a flow are held at VOQs when there is a possibility that they could be delayed at VIMOQs longer than a cell of the same flow that arrives at a later time, and thus preventing out-of-sequence forwarding.

Each IP and CM have a counter, $\sigma_{r,j}$, respectively, to track the occupancy at CMs of cells that are destined to the same OM at $CM(r)$. $\sigma_{r,j}$ is updated each time a cell arrives or departs any VIMOQ for each particular OM. The IP uses the count as a hold-down timer for each VOQ, triggered by $\sigma_{r,j} \geq 1$ as found by a cell from that VOQ at arrival to the CM and notified back to the IP. Here, $\sigma_{r,j}$ is defined as:

$$\sigma_{r,j} = \sum_{i=0}^{i=k-1} \sum_{d=0}^{d=n-1} VIMOQ(i, r, j, d) \tag{4.2}$$

and the hold-down time is $\sigma_{r,j}$ time slots. Therefore, no cell from this VOQ is forwarded to VIMOQs for $\sigma_{r,j}$ time slots, while cells at other VOQs at that IP may still be forwarded.

4.2.5 Flow Control

The switch uses two flow control mechanisms, one flow control mechanism between VIMOQs and IPs and another between CBs and VIMOQs. A VIMOQ has two thresholds it uses for flow control; pause (Tm_p) and resume (Tm_r), where $Tm_p > Tm_r$, in number of cells. When the occupancy of VIMOQ, $|VIMOQ|$, is larger than Tm_p, the VIMOQ signals the corresponding IPs to pause sending cells to it. When the $|VIMOQ| < Tm_r$, the VIMOQ signals the corresponding IPs to resume sending cells to it. Here, Tm_p is such that $C_{VIMOQ} - Tm_p \geq D_{vq}$, where C_{VIMOQ} is the size of the VIMOQ and D_{vq} is the flow-control message delay.

Similar to VIMOQs, CBs have two thresholds; pause (Tc_p) and resume (Tc_r), where $Tc_p > Tc_r$, and Tc_p is such that $C_{CB} - Tc_p \geq D_{cq}$, for a CB size, C_{CB}, and flow-control message delay between a CB and corresponding VIMOQs, D_{cq}. When the occupancy of a CB, $|CB|$, becomes larger than Tc_p, the CB signals the corresponding VIMOQs to pause sending cells to it. When $|CB| < Tc_r$, the CB signals the corresponding VIMOQs to resume forwarding.

4.3 Throughput Analysis

In this section, we analyze the performance of VINE and also prove that it achieves 100% throughput using the concept of queue stability. A switch is defined as stable for a traffic pattern if the queue length is bounded and a switch achieves 100% throughput if it is stable for admissible i.i.d. traffic [57, 58]. With this, we set the following theorem:

Theorem 5. *VINE achieves 100% throughput under admissible i.i.d traffic.*

Proof: Here, we consider the queue to be weakly stable if the drift of the queue occupancy from the initial state is a finite integer $\epsilon \quad \forall\ t$ as $\lim_{t\to\infty}$. Using the definition above, we show that the queue length of VOQs, VIMOQs, and CBs are

weakly stable under i.i.d. traffic, and hence, achieves 100% throughput under that traffic pattern.

Let us represent the queue occupancy of VOQs at time slot t, $\mathbf{N_1}(t)$ as:

$$\mathbf{N_1}(t) = \mathbf{N_1}(t-1) + \mathbf{A_1}(t) - \mathbf{D_1}(t) \tag{4.3}$$

where $\mathbf{A_1}(t)$ is the packet arrival matrix at time slot t to VOQs and $\mathbf{D_1}(t)$ is the service rate matrix of VOQs at time slot t. Solving (4.3) with an initial condition $\mathbf{N_1}(0)$, recursively yields:

$$\mathbf{N_1}(t) = \mathbf{N_1}(0) + \sum_{\gamma=0}^{t} \mathbf{A_1}(\gamma) - \sum_{\gamma=0}^{t} D_1(\gamma) \tag{4.4}$$

Let us consider $d_{1_{u,v}}(t)$ as the service rate received by the VOQ at $IP(u)$ for $OP(v)$ at time slot t or:

$$d_{1_{u,v}}(t) = \begin{cases} \dfrac{1}{N} \leq d_{1_{u,v}}(t) \leq 1 & \text{for } \sigma_{r,j} = 0 \\ \dfrac{1}{\sigma_{r,j}N} \leq d_{1_{u,v}}(t) \leq \dfrac{1}{\sigma_{r,j}} & \text{for } \sigma_{r,j} \geq 1 \end{cases} \tag{4.5}$$

Another way to express $D_1(t)$ is:

$$\mathbf{D_1}(t) = [d_{1_{u,v}}(t)] \tag{4.6}$$

Let us denote the traffic from input ports to the IM stage, $\mathbf{R_1}$, as:

$$\mathbf{R_1} = [\lambda_{u,v}] \tag{4.7}$$

Here, $\lambda_{u,v}$ is the arrival rate of traffic from input u to output v, where

$$u = ik + s \tag{4.8}$$

$$v = jk + d \tag{4.9}$$

and $0 \leq u, v \leq N-1$.

We consider admissible traffic in this analysis, which is defined as:

$$\sum_{u=0}^{N-1} \lambda_{u,v} \leq 1, \ \sum_{v=0}^{N-1} \lambda_{u,v} \leq 1 \tag{4.10}$$

under i.i.d. traffic conditions.

Because $\mathbf{R_1}$ is the aggregate traffic arrival to VOQs it can also be denoted as:

$$\mathbf{R_1} = \sum_{\gamma=0}^{t} \mathbf{A_1}(\gamma) \tag{4.11}$$

Substituting (4.5) into (4.6), and (4.6) and (4.11) into (4.4), yields:

$$\mathbf{N_1}(t) = \begin{cases} \mathbf{N_1}(0) + \mathbf{R_1} - \dfrac{t}{N} * \mathbb{1} & \text{for } \sigma_{r,j} = 0 \\ \mathbf{N_1}(0) + \mathbf{R_1} - \dfrac{t}{\sigma_{r,j} N} * \mathbb{1} & \text{for } \sigma_{r,j} > 1 \end{cases} \tag{4.12}$$

From (4.12), we obtain:

$$\begin{cases} \lim\limits_{t\to\infty} \dfrac{\mathbf{R_1}}{t} - \dfrac{1}{N} * \mathbb{1} \leq \epsilon < \infty & \text{for } \sigma_{r,j} = 0 \\ \lim\limits_{t\to\infty} \dfrac{\mathbf{R_1}}{t} - \dfrac{1}{\sigma_{r,j} N} * \mathbb{1} \leq \epsilon < \infty & \text{for } \sigma_{r,j} > 1 \end{cases} \tag{4.13}$$

From the admissibility condition of $\mathbf{R_1}$, it is easy to see that for any value of t, (4.13) is finite. Hence, from (4.10), (4.12) and (4.13), we conclude that occupancy of VOQ is weakly stable.

■

Now we prove VIMOQs stability. As before, the queue occupancy matrix of VIMOQs at time slot t can be represented as:

$$\mathbf{N_2}(t) = \mathbf{N_2}(t-1) + \mathbf{A_2}(t) - \mathbf{D_2}(t) \tag{4.14}$$

where $\mathbf{A_2}(t)$ is the arrival matrix at time slot t to VIMOQs and $\mathbf{D_2}(t)$ is the service rate matrix of VOMQs at time slot t. Solving (4.14) recursively with consideration

of an initial condition for $\mathbf{N_2}(t)$, yields:

$$\mathbf{N_2}(t) = \mathbf{N_2}(0) + \sum_{\gamma=0}^{t} \mathbf{A_2}(\gamma) - \sum_{\gamma=0}^{t} \mathbf{D_2}(\gamma) \tag{4.15}$$

Because VIMOQs are serviced using round-robin, this means a VIMOQ is serviced at least once every N time slots. The service rate of the VIMOQs at time slot t, $d_{2_{\mu,v}}(t)$ is:

$$\frac{1}{N} \leq d_{2_{\mu,v}}(t) \leq 1$$

Then, the service matrix of VOMQs is:

$$\mathbf{D_2}(t) = [d_{2_{\mu,v}}(t)] \tag{4.16}$$

Let us represent the aggregate traffic arrival to VIMOQs as $\mathbf{R_2}$, which is the traffic directed towards OMs and it is derived from $\mathbf{R_1}$ and the permutations of IMs. The configuration of IMs at time slot t that connects $IP(i,s)$ to $L_I(i,r)$ are represented as an $N \times N$ permutation matrix, $\mathbf{\Pi}(t) = [\pi_{u,v}]$ and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = rk + i \\ 0 & \text{elsewhere.} \end{cases}$$

The configuration of IMs can be represented as a compound permutation matrix, $\mathbf{P_1}$, which is the sum of IMs permutations over k time slots is:

$$\mathbf{P_1} = \sum^{k} \mathbf{\Pi}(t) \tag{4.17}$$

Because the configuration is repeated every k time slots, the traffic load from the same input going to each VIMOQ is $\frac{1}{k}$ of the traffic load of $\mathbf{R_1}$. Therefore, a row of $\mathbf{R_2}$ is the sum of the row elements of $\mathbf{R_1}$ at the non zero positions of $\mathbf{P_1}$, normalized

by k. This is:

$$\mathbf{R_2} = \frac{1}{k}((\mathbf{R_1} * \mathbb{1}) \circ \mathbf{P_1}) \tag{4.18}$$

Because $\mathbf{R_2}$ is the aggregate traffic arrival to VIMOQs it can also be denoted as:

$$\mathbf{R_2} = \sum_{\gamma=0}^{t} \mathbf{A_2}(\gamma) \tag{4.19}$$

Because $\mathbf{R_1}$ is admissible, $\mathbf{R_2}$ would be admissible. Let us assume that VIMOQs are serviced at least once every N time slots. Hence, $d_{2_{\mu,v}} = \frac{1}{N}\ \forall\ \mu$ and v in (4.16). Substituting (4.16) and (4.19) into (4.15) gives:

$$\mathbf{N_2}(t) = \mathbf{N_2}(0) + \mathbf{R_2} - \frac{1}{N}\mathbf{P_1} \tag{4.20}$$

From (4.20), we get:

$$\mathbf{R_2} - \frac{1}{N}\mathbf{P_1} \leq \epsilon < \infty \tag{4.21}$$

Recalling that $\mathbf{R_2}$ is admissible and by substituting $\mathbf{P_1}$ and $\mathbf{R_2}$ into (4.21), it is easy to see that ϵ is finite. Hence, from (4.20) and (4.21), we conclude that the occupancy of VIMOQ is weakly stable.

■

Now we prove the stability of CBs. The queue occupancy matrix of CBs at time slot t can be represented as:

$$\mathbf{N_3}(t) = \mathbf{N_3}(t-1) + \mathbf{A_2}(t) - \mathbf{D_3}(t) \tag{4.22}$$

where $\mathbf{A_3}(t)$ is the packet arrival matrix at time slot t CBs, and $\mathbf{D_3}(t)$ is the service rate matrix of CBs at time slot t. Solving (4.22) recursively as before yields:

$$\mathbf{N_3}(t) = \mathbf{N_3}(0) + \sum_{\gamma=0}^{t} \mathbf{A_3}(\gamma) - \sum_{\gamma=0}^{t} \mathbf{D_3}(\gamma) \tag{4.23}$$

Because a CB is serviced at least once every k time slots. Hence, the service rate of the CB at $OP(v)$ at time slot t, $d_{3_v}(t)$ is:

$$\frac{1}{k} \leq d_{3_v}(t) \leq 1$$

and service matrix of CBs is:

$$\mathbf{D_3}(t) = [d_{3_v}(t)] \tag{4.24}$$

We define a $k \times N$ matrix, $\mathbf{D_s}$, built by concatenating k $k \times n$ diagonal matrices, $\mathbf{D}$, as: $\mathbf{D_s}$ is an $m \times N$ matrix, built by concatenating N $k \times 1$ vector of all ones, $\vec{1}$, as:

$$\mathbf{D_s} = [\vec{1}, \cdots, \vec{1}] \tag{4.25}$$

We define a $1 \times k$ row vector, $\vec{A}$, built by setting the first element to 1 and every other element to 0, or:

$$\vec{A} = [1 \cdots 0] \tag{4.26}$$

$\vec{A_s}$ is an $N \times 1$ column vector, built by concatenating k $\vec{A}$ and taking its transpose, or:

$$\vec{A_s} = [\vec{A_{s_1}}, \cdots, \vec{A_{s_k}}]^T \tag{4.27}$$

where $\vec{A_{s_1}} = \vec{A_{s_k}} = \vec{A}$, such that

$$\vec{A_s} = [\vec{A}, \cdots, \vec{A}]^T \tag{4.28}$$

The traffic queued at the CB of an OP, $\mathbf{R_3}(v)$, is the multiplication of $\mathbf{D_s}$, $\mathbf{R_2}(j, d)$, and a vector of all ones, $\vec{A_s}$, or:

$$\mathbf{R_3}(v) = \mathbf{D_s} * \mathbf{R_2}(j, d) * \vec{A_s} \tag{4.29}$$

Because $\mathbf{R_2}$ is admissible, $\mathbf{R_3}$ is also admissible.

Hence, aggregate traffic arrival to the CB can also be written as:

$$\mathbf{R_3} = \sum_{\gamma=0}^{t} A_3(\gamma) \tag{4.30}$$

Let us assume $d_{3_v}(t) = \frac{1}{k}\ \forall\ v$ in (4.24), which is the worst case scenario at which a CB gets serviced once every k time slots. Substituting (4.24) and (4.30) into (4.23):

$$\mathbf{N_3}(t) = \mathbf{N_3}(0) + \mathbf{R_3} - \frac{1}{k} * \vec{1} \tag{4.31}$$

where

$$\mathbf{R_3} - \frac{1}{k} * \vec{1} \leq \epsilon < \infty \tag{4.32}$$

With $\mathbf{R_3}$ being admissible, and by substituting $\mathbf{R_3}$ into (4.32), it is easy to see that ϵ is finite. Hence, from (4.31) and (4.32), we conclude that the occupancy of CB is also weakly stable.

■

This completes the proof of Theorem 5.

■

4.4 Delay Analysis

In this section, we analyze the delay a cell experiences while traversing VINE and also show that this delay approximately equal to the delay the cell experiences while traversing an OQ switch for any admissible i.i.d. traffic through the following theorem. Table 4.2 lists the definition of terms used in the discussion

Theorem 6. *The difference between the maximum delay a cell experiences while traversing VINE and an OQ switch is infinitesimal under admissible i.i.d. traffic.*

We selected two traffic scenarios that causes cells to contend for $L_C(r, j)$ towards the destination OM, stresses the occupancy CBs of VINE, and also shows the advantage of OQ's speedup.

Table 4.2 Notations used in the Delay Analysis of VINE

$c_{y,\tau}$	The τth cell of flow y from $IP(i, s)$ to $OP(j, d)$.
$t_{a_{1,\tau}}$	Arrival time of $c_{y,\tau}$ in $VOQ(i, s, j, d)$ at $IP(i, s)$ of VINE.
$t_{a_{2,\tau}}$	Arrival time of $c_{y,\tau}$ in $VIMOQ(r, i, j, d)$ of VINE.
$t_{a_{3,\tau}}$	Arrival time of $c_{y,\tau}$ in $CB(r, j, d)$ of VINE.
t_{dv_τ}	Departure time of $c_{y,\tau}$ from $CB(r, j, d)$ of VINE.
$t_{d_{mv_\tau}}$	Maximum departure time of $c_{y,\tau}$ from $CB(r, j, d)$ of VINE.
W_{V_τ}	Waiting time of cell $c_{y,\tau}$ at $CB(r, j, d)$ of VINE.
R_τ	Residual waiting time of $c_{y,\tau}$ at $CB(r, j, d)$ of VINE.
S_t	Expected service time of $c_{y,\tau}$ at $CB(r, j, d)$ of VINE.
t_{I_τ}	Arrival time of $c_{y,\tau}$ at $IP(u)$ of OQ switch.
t_{O_τ}	Arrival time of $c_{y,\tau}$ at $OP(v)$ of OQ switch.
t_{do_τ}	Departure time of $c_{y,\tau}$ at $OP(v)$ of OQ switch.
W_{O_τ}	Waiting time of cell $c_{y,\tau}$ at $OP(v)$ of OQ switch.
$t_{d_{mo_\tau}}$	Maximum Departure time of $c_{y,\tau}$ from $OP(v)$ of OQ switch.

We prove this theorem using lemmas.

Lemma 1. *Considering a hot-spot traffic pattern, where all IPs send a cell to the same OP. The difference between the maximum delay a cell experiences while traversing VINE and an OQ switch is one time slot.*

Proof: We assume that there are previously no cells in the switch. Firstly we consider VINE, let the arrival time of cell, $c_{y,\tau}$, to a VOQ, $t_{a_{1,\tau}}$, be:

$$t_{a_{1,\tau}} = t_x \tag{4.33}$$

Because the VOQs were initially empty, the arrival time of cells to VIMOQs, $t_{a_{2,\tau}}$, is:

$$t_{a_{2,\tau}} = t_x + 1 \tag{4.34}$$

Also because VIMOQs were initially empty, the arrival time of cells to the CB, $t_{a_{3,\tau}}$, is:

$$t_{a_{3,\tau}} = t_x + 1 + \theta_\tau = t_{a_{2,\tau}} + \theta_\tau \tag{4.35}$$

where β_τ is the position of the cell based on the round-robin arbitration at the CM and $1 \leq \beta_\tau \leq k$. The departure time of a cell $c_{y,\tau}$ from the CB is:

$$t_{dv_\tau} = t_{a_{3,\tau}} + W_{V_\tau} \tag{4.36}$$

$$W_{V_\tau} = R_\tau + S_t \tag{4.37}$$

where R_τ is the time it takes for the cell ahead of $c_{y,\tau}$ at the CB to be forwarded out the OP and $S_t = 1$. Because $1 \leq \beta_\tau \leq k$ and there are N source IPs based on the traffic scenario, the maximum number of cells ahead of $c_{y,\tau}$ at the CB upon arrival is:

$$N - k$$

Therefore, the waiting time of $c_{y,\tau}$, W_{V_τ}, is

$$W_{V_\tau} = (N - k) + 1 \tag{4.38}$$

Substituting (4.38) into (4.36) yields:

$$t_{dv_\tau} = t_{a_{3,\tau}} + (N - k) + 1 \tag{4.39}$$

From (4.34) and (4.35), the maximum departure time from the CB is:

$$t_{d_{mv_\tau}} = t_x + 2 + N \tag{4.40}$$

Let us now consider the OQ switch. Let the arrival time of cell, c_τ, to IPs, t_{I_τ}, be:

$$t_{I_\tau} = t_x \tag{4.41}$$

Because OQ switch uses speedup, the arrival time of cells to the output port queue, t_{O_τ}, is:

$$t_{O_\tau} = t_x + 1 \tag{4.42}$$

The departure time of a cell $c_{y,\tau}$ from the OP queue is:

$$t_{do_\tau} = t_{O_\tau} + W_{O_\tau} \tag{4.43}$$

where $1 \leq W_{O_\tau} \leq N$.

Therefore, the maximum departure time from the OP queue is:

$$t_{d_{mo_\tau}} = t_{O_\tau} + N = t_x + 1 + N \tag{4.44}$$

From (4.40) and (4.44), we see that the difference in the maximum delay observed by a cell while traversing VINE or OQ under the hot-spot traffic is one time slot.

■

Lemma 2. *Considering a traffic pattern where all IPs in the same IM send a cell to different IPs in the same OM and no other traffic traverses the switch. The difference between the maximum delay a cell experiences while traversing VINE and an OQ switch is one time slot.*

Firstly we consider VINE. Let the arrival time of cell, $c_{y,\tau}$, to a VOQ, $t_{a_{1,\tau}}$, be:

$$t_{a_{1,\tau}} = t_x \tag{4.45}$$

Because the VOQs were initially empty, the arrival time of cells to VIMOQs, $t_{a_{2,\tau}}$, is:

$$t_{a_{2,\tau}} = t_x + 1 \tag{4.46}$$

Also since VIMOQs were initially empty, the arrival time of cells to the CB, $t_{a_{3,\tau}}$, is:

$$t_{a_{3,\tau}} = t_{a_{2,\tau}} + 1 = t_x + 2 \tag{4.47}$$

The departure time of a cell $c_{y,\tau}$ from the CB is:

$$t_{d_\tau} = t_{a_{3,\tau}} + 1 \tag{4.48}$$

Therefore, the maximum departure time from the CB is:

$$t_{d_{mv_\tau}} = t_x + 3 \tag{4.49}$$

Let us now consider the OQ switch. For the OQ switch the traffic pattern is between n IPs and OPs. Let the arrival time of cell, $c_{y,\tau}$, to IPs, t_{I_τ}, be:

$$t_{I_\tau} = t_x \tag{4.50}$$

The arrival time of cells to the output port queue, t_{O_τ}, is:

$$t_{O_\tau} = t_x + 1 \tag{4.51}$$

The departure time of a cell $c_{y,\tau}$ from the OP queue is:

$$t_{do_\tau} = t_{O_\tau} + 1 \tag{4.52}$$

Therefore, the maximum departure time from the OP queue is:

$$t_{d_{mo_\tau}} = t_x + 2 \tag{4.53}$$

From (4.49) and (4.53), we see that the difference in the maximum delay experienced by a cell while traversing VINE or OQ under the above traffic pattern is one time slot.

■

This completes the proof of Theorem 6.

■

4.5 Performance Analysis

The performance analysis of VINE are obtained through computer simulation under uniform and non uniform traffic models. The performance of VINE is compared with that of an output-queued (OQ), an Memory-Memory-Memory Clos-network (MMM) switch, and MMM switch with extended memory (MM^{e}M). We have chosen these switches for comparison because the OQ switch is an ideal switch, while MMM and MM^{e}M achieve the highest performance among Clos-network switches, despite been categorized as different architectures. We considered switches with size $N = \{64, 256\}$.

4.5.1 Uniform Traffic

Figures 4.2 and 4.3 show the average delay under uniform traffic with Bernoulli arrivals for $N = 64$ and $N = 256$, respectively. The figures show VINE and OQ experiences the same average delay, which is consistent with the claims in Section 4.4. The figures also show that the high performing MMM and MM^{e}M switches experiences delay higher than VINE. This high performance is attributed to the unique switch architecture, efficient load-balancing, and in-sequence mechanisms. Our simulator can only simulate small MM^{e}M switches for queueing analysis due

to their high memory requirement. So we simulated the switch under uniform traffic pattern for $N = 64$.

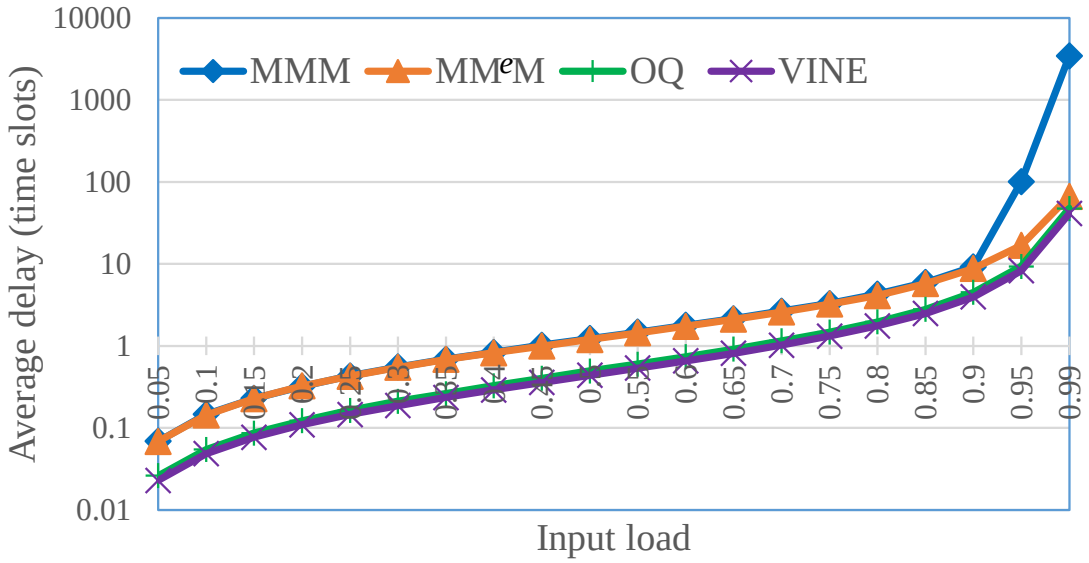


Figure 4.2 Average queueing delay under uniform traffic for N=64.

4.5.2 Nonuniform Traffic

We also evaluated the throughput of VINE, MMM, MM^{e}M under unbalanced [74] and diagonal [63] patterns as nonuniform traffic. Figure 4.4 shows the performance of VINE, MMM, and MM^{e}M under unbalanced traffic for unbalanced probability $0 \leq w \leq 1$. The results show that VINE attains 100% throughput, outperforming MMM and sharing similar performance to that of MM^{e}M, however, with lower complexity and while forwarding cells in sequence. One should note that MM^{e}M forward cells out of sequence. Figure 4.6 shows that the average delay of VINE under unbalanced traffic for $N = 256$. The result shows that VINE experiences the same average delay as the high-performing OQ switch. Figure 4.5 shows the throughput of VINE, MMM, and MM^{e}M switches under diagonal traffic for diagonal probability $0 \leq g \leq 1$. Also the figure shows that VINE attains 100% throughput, outperforms MMM, and matches

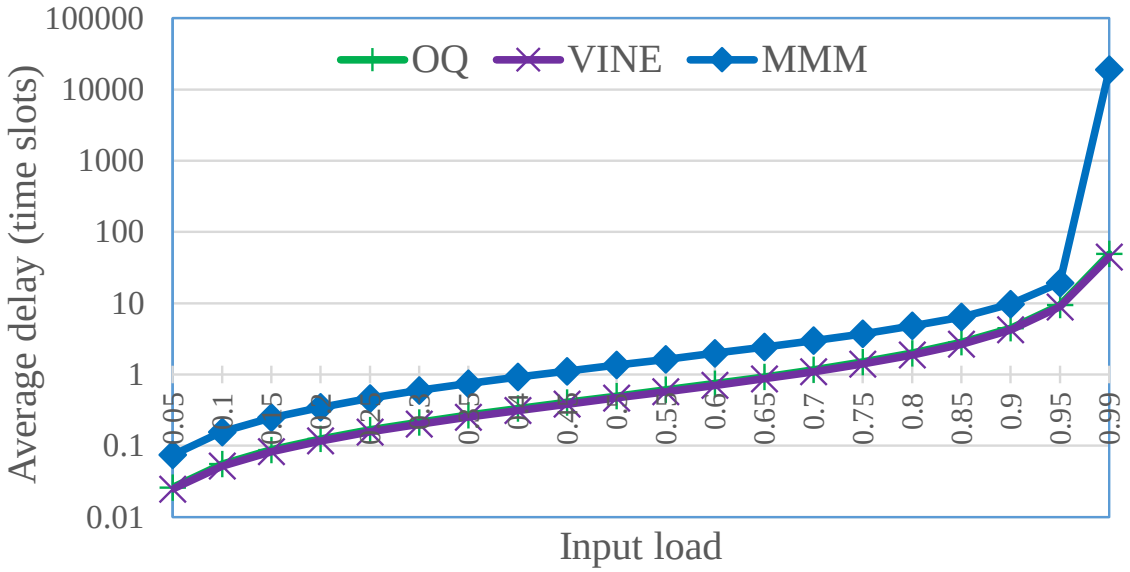


Figure 4.3 Average queueing delay under uniform traffic for N=256.

MM^{e}M under this traffic pattern. These results are the product of the load-balancing operation in VINE.

4.6 Conclusions

We introduced a scalable and high-performing load-balancing Clos-network switch that uses virtual input-module output queues at CMs and crosspoint buffers at OMs. The IMs are bufferless modules and the OMs are buffered ones. Irrespective of IMs being bufferless, the switch does not use module or port matching. Instead the switch uses a low-complexity configuration scheme, where IMs follow a predeterministic configuration ($O(1)$), while IP, CM, and OM links use round-robin arbitration. We adopted an in-sequence mechanism that operates at the inputs of VINE that ensures forwarding of cells in-sequence. The switch architecture, load-balancing, and in-sequence mechanisms makes VINE outperform the MMM switches, while performing like an OQ switch. We modeled and analyzed the performance of VINE and showed that for admissible independent and identically distributed traffic, the

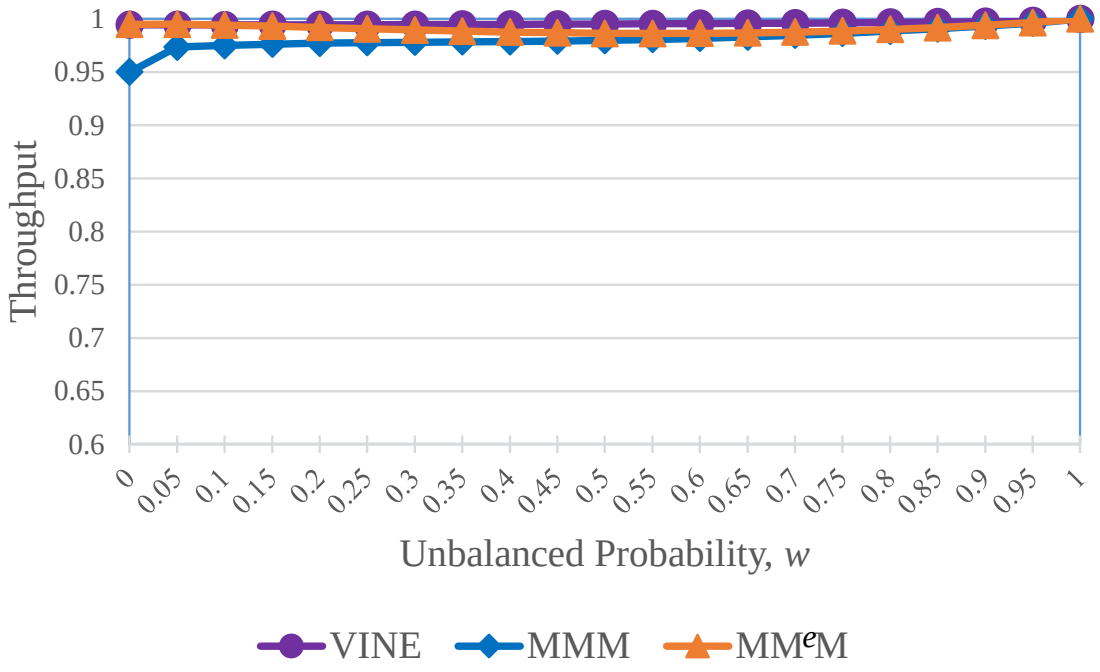


Figure 4.4 Throughput under unbalanced traffic for N=256.

switch achieves 100% throughput and performs like an OQ switch. This high performance is achieved without switch expansion nor speepdup. We showed, through computer simulation, that for all tested traffic, the switch achieves 100% throughput for uniform and nonuniform traffic distributions.

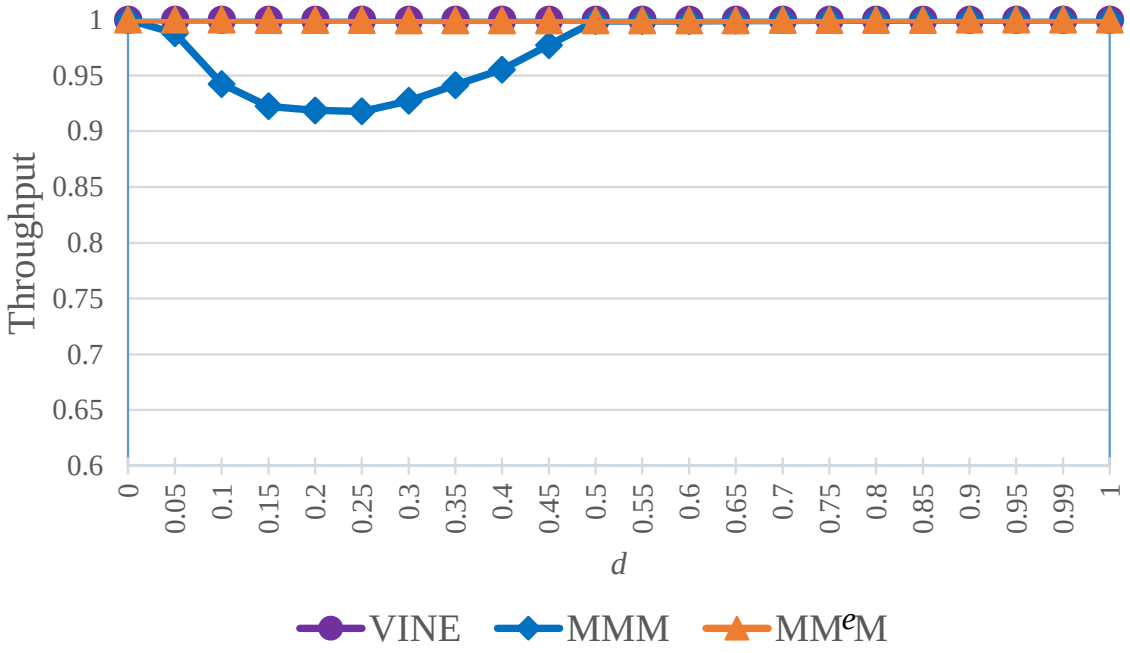


Figure 4.5 Throughput under diagonal traffic for N=256.

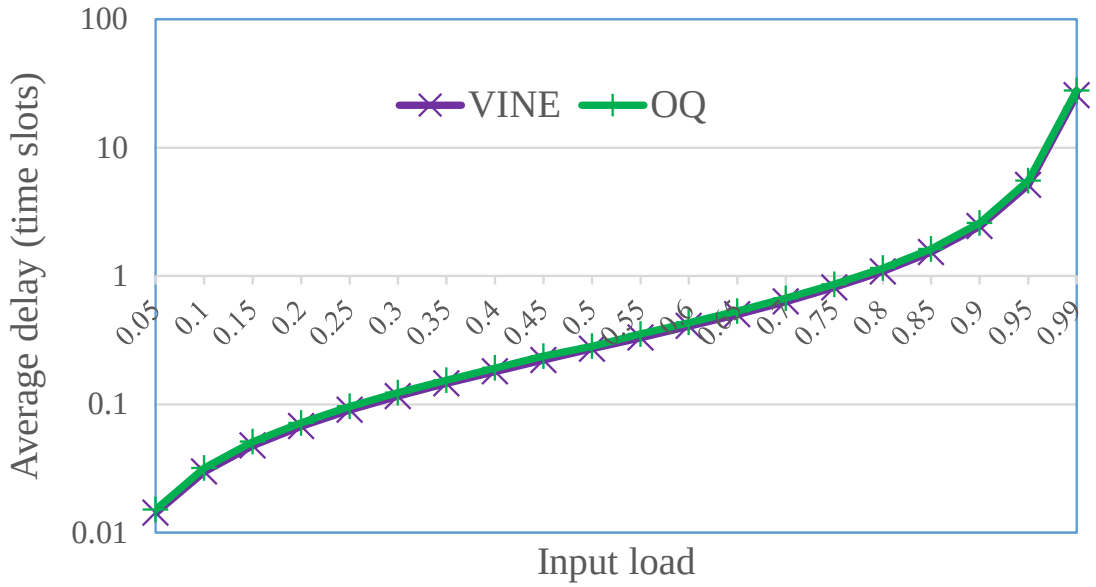


Figure 4.6 Unbalanced traffic, $w = 0.65$ and N=256.

CHAPTER 5

MODELING OF NETWORK PACKET SWITCHES USING MATRIX ANALYSIS

5.1 Introduction

The operation of a packet switch (or just switch for brevity) is to forward Internet packets from inputs to outputs. An $N \times N$ switch may have its inputs indexed by $0 \leq u \leq N-1$ and its outputs by $0 \leq v \leq N-1$. Different packet switch architectures require different configuration schemes. These schemes have a direct effect on the operation of switching traffic from inputs to outputs and, therefore, on the performance of the switch. Performing switch analysis is a critical tool to which designers resort to determine the worthiness of a switch architecture. Moreover, these tools can be specifically targeted at identifying a performance metric, such as stability, throughput, or delay among the most demanded. However, analyzing the different performance metrics of the switch also requires different tools, some of which sometimes come at the expense of high complexity or are rendered complex to apply.

Here, we consider that a packet switch may receive variable length Internet packets and segment them into fixed-size cells for internal switching, and re-assembling them before packets depart the switch. In this way, the time it takes to switch a cell from inputs to outputs is also fixed, and it is referred to a time slot.

The performance of a switch may be determined by different metrics. Some of the more sought after are:

Throughput. In general, throughput of a switch is conveniently represented as a normalized estimate; that is, the ratio of the number of cells that left the outputs of the switch to the arrived cells at the inputs of the switch. The throughput of a

switch is one the the most important performance metrics as it enables us to evaluate other parameters, such a delay. The delay of a cell (or packet) can be evaluated only if the throughput of a switch for a given input load is full; that is, the number of cells leaving the switch equals that entering the switch. The throughput of a switch may be estimated by computer simulation, probabilistic analysis, direct estimation, or tight or loose bound estimation for cases where absolute estimation is complex [1, 2, 25, 36, 37, 46, 49, 66, 69]. In the particular case 100% throughput is expected under 100% input load (full load that inputs may sustain), stability of the queues, and therefore, the switch may be analyzed as binary test [20, 32, 55–58, 67].

Stability. A switch is considered to be stable if for any admissible input traffic, the switch is capable of forwarding the incoming traffic to the respective output(s), such that the queue occupancy of the switch does not grow infinitely.

We define admissible traffic as

$$\sum_{u=0}^{N-1} \lambda_{u,v} \leq 1, \ \sum_{v=0}^{N-1} \lambda_{u,v} \leq 1 \tag{5.1}$$

where, $\lambda_{u,v}$ is the arrival rate of traffic from input u to output v of a switch. Such admissibility is considered here under independent and identically distributed (i.i.d.) traffic conditions.

In this case, the queue occupancy is not expected to grow if the input is admissible. However, some queueing may occur for short periods of time but as long as the average profile of the input load remains admissible, queues are not expected to grow infinitely, should input buffers have sufficient capacity [55–58]. The finite queue occupancy of a switch can be evaluated by computer simulation [59], modeling the queueing system of the switch [65], by using a fluid model to determine its stability [20], or by using a stability criteria such as Lyapunov analysis. For a switch with queue occupancy matrix $\mathbf{L}(n)$ at time n, state vector

$$Y(n) = (\mathbf{L}(n), X(n)) \tag{5.2}$$

where $X(n)$ is an integer vector, and a Lyapunov function

$$V(L(n)) = L(n)\ Z\ L(n)^T \tag{5.3}$$

If there is a symmetric copositive. An N×N matrix B is copositive if $L\mathrm{B}L^T \geq 0\ \forall\ L \in \mathbb{R}^{+N}$ matrix $Z \in \mathbb{R}^{N\times N}$, and two positive numbers $\epsilon \in \mathbb{R}^+$ and $U \in \mathbb{R}^+$, such that:

$$E[V(L(n+1) - V(L(n)|L(n)] < -\epsilon||L(n)||\ \ \forall\, Y(n) :$$

$$||L(n)|| > U \tag{5.4}$$

the switch is considered to be stable [3, 51, 55–58, 68, 75].

Queueing delay. The queueing delay, or just delay, is the time a cell waits in a queue before being forwarded to the destination output. This waiting time may depend on both the presence of other cells in other queues contending to be forwarded to the output and on the scheme used to select the next cell to be forwarded. Then, the average queueing delay of a switch is the average estimate on the cells passing through the switch. For analysis of a switch under i.i.d. traffic, the average switch delay of switch is equivalent to that experienced by traffic coming through one of the outputs.

The average queueing delay of a switch may be measured through computer simulation or by modeling the queues as a Markov process model and analyzing the model, or queueing analysis [4, 10, 34, 36, 51, 52, 61, 65, 75]. However, the last option is difficult to adapt for various switch architectures, traffic types, and selection schemes.

These issues raise the question, can the throughput of a switch be analyzed using a simpler method that provides insight into the switching fabric and numeric throughput values?

We propose the use of matrix analysis as a tool for analyzing the operation of a switch on the incoming traffic and thus, to analyze the performance of a switch. Matrix analysis provides a deep insight into the operation of the switching fabric, and also with this analysis we may obtain a numeric throughput.

The use of matrix decomposition as a configuration scheme, has been previously applied to a few switching architectures. In [43], a matrix decomposition algorithm was presented to route cells in a rearrangeable three-stage Clos network by performing a row-wise matrix decomposition. In [7, 8, 45], a scheduling algorithm that uses the results from Birkhoff decomposition [5] and von Neumann algorithms [79] for a doubly stochastic matrix was proposed for an input-buffered crossbar switch.

Different from those applications, our objective is to apply matrix analysis in modeling the operation of the switch on the incoming traffic and to estimate the performance of the switch. In this chapter, we show that matrix analysis can be applied to both single and multi-stage switch architectures that use either pre-determined or random arbitration schemes.

The remainder of this chapter is organized as follows: Section 5.2, introduces our matrix analysis approach in general terms. Section 5.3, describes several application cases, from single to multi-stage switches that may use load-balancing functions or other configuration schemes. In this examples, we estimate the switch throughput or verify that a switch achieves 100% throughput. Section 5.4, presents our conclusions.

5.2 Throughput Analysis through Matrix Analysis

For any single or multistage switching architecture, the internal configuration of the switch can be represented by a compound permutation $\mathbf{P}$, or a set of permutations $\mathbf{P_1}, \cdots, \mathbf{P}_{\gamma-\mathbf{1}}$. where γ is the number of stages in the multistage architecture and $\gamma > 1$.

Let's consider that the traffic incoming to a switch, or input matrix $\mathbf{R_1}$, is defined as:

$$\mathbf{R_1} = [\lambda_{u,v}] \tag{5.5}$$

where $\mathbf{R_1}$ is admissible.

The traffic at the output port of a single-stage switch, $\mathbf{R_2}$, is:

$$\mathbf{R_2} = \mathbf{R_1}\alpha\mathbf{P} \tag{5.6}$$

where α is defined by the configuration of the switch and $\mathbf{P}$ is the connection provided by the configuration. Moreover, for a multistage switch architecture with γ stages, the matrix at the output port, $\mathbf{R}_\gamma$, is:

$$\mathbf{R}_\gamma = (((\mathbf{R_1}\alpha\mathbf{P_1})\alpha\mathbf{P_2})\cdots\alpha\mathbf{P}_{\gamma-\mathbf{1}}) \tag{5.7}$$

The throughput of a single-stage or multi-stage switch with input $\mathbf{R_1}$ and output $\mathbf{R_2}$ or $\mathbf{R}_\gamma$, respectively, is calculated by dividing the sum of the columns in $\mathbf{R_2}$ or $\mathbf{R}_\gamma$ by the sum of the columns in $\mathbf{R_1}$ and adding up the results. For an output matrix, $\mathbf{R_O}$, or:

$$\mathbf{R_O} = [\lambda_{\mu,v}] \tag{5.8}$$

where $\mathbf{R_O} = \mathbf{R_2}$ for a single-stage and $\mathbf{R_O} = \mathbf{R}_\gamma$ for a multi-stage switch, the throughput of the switch is calculated by:

$$\text{Throughput} = \sum \left(\frac{\sum_{\mu=0}^{N-1} \lambda_{\mu,v} | v = 0, \cdots, N-1}{\sum_{u=0}^{N-1} \lambda_{u,v} | v = 0, \cdots, N-1} \right) \tag{5.9}$$

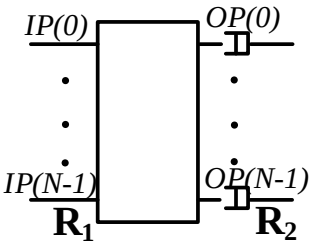
5.3 Application Examples of Matrix Analysis on Switch Architectures

In this section, we apply this technique to an output queued (OQ), input-queued, a load-balancing Birkhoff-von-Neumann (LBvN) switches, which can be classified as single (e.g., OQ and IQ) and multi-stage switches (e.g., LBvN and MM^{e}M), with deterministic (e.g., LBvN) and non-deterministic configuration patterns. These switches are selected because of their high performance and herein, we show that the proposed approach can be used to estimate the switch throughput. The architecture of the switches used in the examples are shown in Figure 5.1.

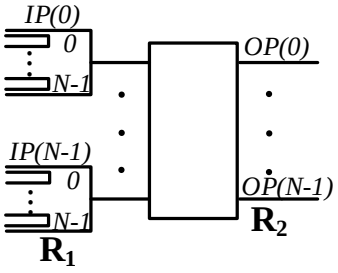
5.3.1 Output-queued (OQ) Switch

In this section, we analyze the performance of an output-queued (OQ) switch. An OQ switch has queues at the outputs. Figure 5.1(a) shows the architecture of an OQ switch. Output queueing enables forwarding multiple cells at the same time to the same output, where they are queued before being forwarded out of output port [18, 30, 38]. However, the queues of an OQ switch operate at N times the speed of the line rate [15, 21, 38, 53]. When the line rate is high or the switch size is large, implementing this speedup of the queues becomes infeasible [6, 14, 55, 62, 77]. The OQ switch is considered an ideal switch; the throughput is 100% under any admissible traffic pattern and cells are available for leaving the switch right after arrival. However, the order in which cells leave an output depends on the forwarding policy applied at the output queues. By using matrix analysis, we show that the OQ switch effectively achieves 100% throughput.

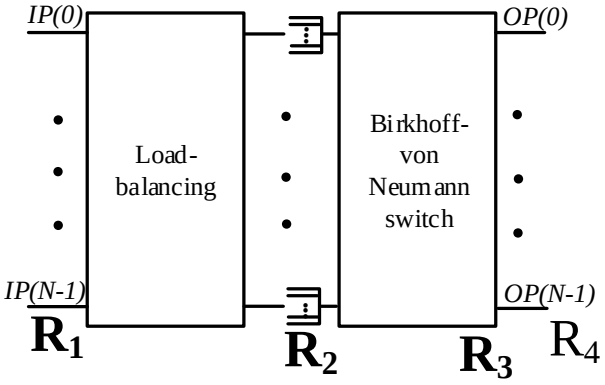
Let us denote the traffic coming to the input ports and then leaving to the output ports of the OQ switch as $\mathbf{R_1}$, $\mathbf{R_2}$, respectively. Here, $\mathbf{R_1}$ and $\mathbf{R_2}$ are $N \times N$ matrices.



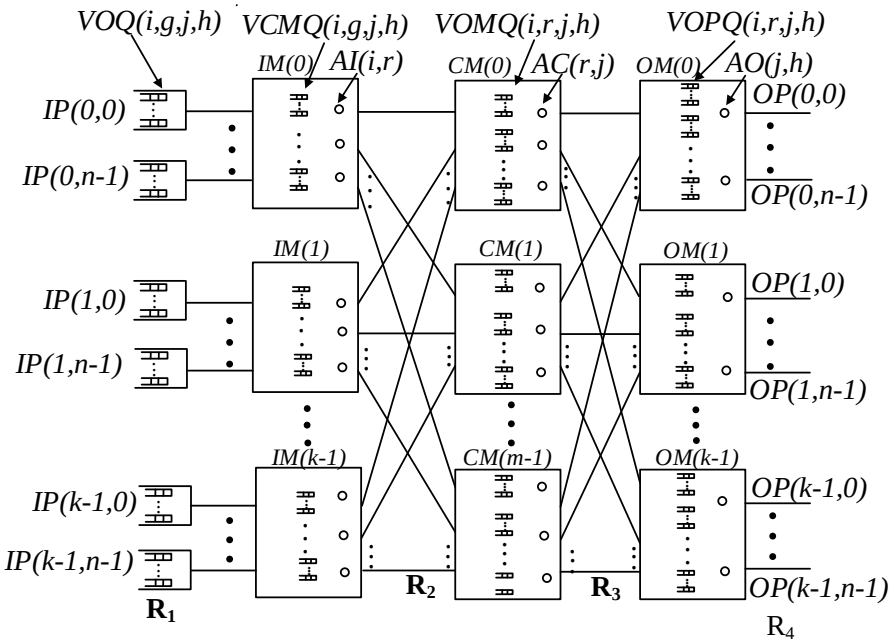
(a) OQ switch



(b) IQ switch



(c) LBvN switch



(d) MM^{e}M switch

Figure 5.1 Switch architectures of example switches.

Here, $\mathbf{R_1}$ is obtained from (5.5), the configuration of the OQ switching fabric at time slot t that connects $IP(i)$ to $OP(j)$ is represented as an $N \times N$ permutation matrix, $\mathbf{\Pi}(t) = [\pi_{u,v}]$, and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{if } u \text{ connects to } v \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

The configuration of the OQ switch can be represented as a compound permutation matrix, $\mathbf{P}$, as follows,

$$\mathbf{P} = \sum_{t} \mathbf{\Pi}(t) \tag{5.11}$$

While $\mathbf{R_2}$ is:

$$\mathbf{R_2} = (\mathbf{R_1} \circ \mathbf{P}) \tag{5.12}$$

where $\circ$ denotes element/position wise multiplication.

As example, we calculate the throughput for a 4×4 OQ switch for any input traffic pattern. Let the input traffic matrix be:

$$\mathbf{R_1} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix}$$

From (5.11), the compound permutation matrix of the OQ switch is:

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Using (5.12), we get $\mathbf{R_2}$, the traffic matrix at the output ports:

$$\mathbf{R_2} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix} \tag{5.13}$$

From (5.13), we see that $\mathbf{R_1} = \mathbf{R_2}$ and this is achieved over one time slot, which implies a speedup of N. By using (5.9), it is easy to see that the OQ switch attains 100% throughput under admissible traffic and this result is equivalent to the well-known performance of this switch.

5.3.2 Input-queued (IQ) Switch with iterative Round Robin Matching (iRRM)

In this section, we analyze an input queue (IQ) switch which uses iterative round robin matching (iRRM) as the configuration scheme [54]. An IQ switch has queues at the inputs and usually virtual output queues (VOQs), which are used to queue cells for each destination output. Figure 5.1(b) shows the architecture of an IQ switch. iRRM works as follows: At time slot t, all the unmatched IPs with one or more queued cells for an OP sends a request to the OP.

The OP chooses the request that appears next in a round-robin schedule starting from the OP with the highest priority, and notifies each requesting IP whether the request is granted. The round-robin pointer of the OP is updated to one location beyond the granted IP (modulo u).

The IP accepts the grant from an OP in a round-robin schedule starting from the OP with the highest priority and updates the round-robin pointer to one location beyond the accepted OP (modulo v).

Let us denote the traffic coming to the input and the output ports of the IQ switch as $\mathbf{R_1}$, $\mathbf{R_3}$, respectively, as $N \times N$ matrices.

The traffic matrix at the input port is obtained from (5.5), the configuration of the IQ switch at time t can be represented as an $N \times N$ permutation matrix, $\mathbf{\Pi}(t) = [\pi_{u,v}]$, and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{if } u \text{ connects to } v \\ 0 & \text{otherwise.} \end{cases} \tag{5.14}$$

Because each IP connects to an OP in a round-robin fashion under the presence of traffic for that OP at the IP, the configuration of the switch can be represented as a pre-deterministic sequence of periodic permutations.

$$\mathbf{P}(t) = \sum \mathbf{\Pi}(t) \tag{5.15}$$

A permutation then indicates the connections from an IP to the OP for which it has traffic. This permutation may change as a pattern, in a round-robin schedule, each time slot.

The traffic matrix at the output ports after each time slot t, $\mathbf{R_P}$(t), is:

$$\mathbf{R_P}(t) = \mathbf{R_P}(t-1) - \alpha_t \mathbf{P}(t) \tag{5.16}$$

where α_t is the smallest weight element of $\mathbf{R_P}(t)$ at non-zero positions of $\mathbf{P}(t)$. In (5.16), $\mathbf{R_P}(t=0)$ is the input matrix $\mathbf{R_1}$. Therefore, the traffic matrix at the output ports after N time slots, or the matrix for the matched connections, $\mathbf{R_3}$, is:

$$\mathbf{R_2} = \mathbf{R_1} - \mathbf{R_P}(t=N) \tag{5.17}$$

where $\mathbf{R_P}(t=N)$ is the matrix of unmatched connections.

5.3.2.1 Example of an IQ Switch with iRRM under Uniform Traffic In this example, we consider a uniformly distributed input traffic for a 4×4 IQ switch. Let the input traffic matrix be:

$$\mathbf{R_1} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix}$$

$\mathbf{R_1}$ is decomposed into $\mathbf{R_1}(t)$ for each time slot t. From (5.15) and (5.16), we obtain:

$$\mathbf{R_P}(1) = \begin{bmatrix} 0 & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & 0 & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & 0 & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & 0 \end{bmatrix}$$

$$\mathbf{R_P}(2) = \begin{bmatrix} 0 & \lambda_{0,1} & \lambda_{0,2} & 0 \\ 0 & 0 & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & 0 & 0 & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & 0 & 0 \end{bmatrix}$$

$$\mathbf{R_P}(3) = \begin{bmatrix} 0 & \lambda_{0,1} & 0 & 0 \\ 0 & 0 & \lambda_{1,2} & 0 \\ 0 & 0 & 0 & \lambda_{2,3} \\ \lambda_{3,0} & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R_P}(4) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $\mathbf{R_P}(4)$ is the matrix for unmatched connections, but by being a null matrix, it indicates that all connections are matched.

Using (5.17), we obtain $\mathbf{R_2}$, the traffic matrix at the output ports after N time slots, or:

$$\mathbf{R_2} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix} \tag{5.18}$$

As stated above, each IP connects to each OP once every N time slots. Because $\mathbf{R_1}$ is uniformly distributed, each IP forwards traffic at a rate of $\frac{1}{N}$, which for this example, it is represented as:

$$\mathbf{R_1} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix} \tag{5.19}$$

Therefore,

$$\mathbf{R_2} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix} \tag{5.20}$$

From (5.18) and (5.20), we see that $\mathbf{R_1} = \mathbf{R_2}$. By using (5.9), it is clear that this switch achieves 100% throughput under uniform traffic.

5.3.2.2 Example of an IQ Switch with iRRM under Nonuniform Traffic

In this example, we consider a nonuniform input matrix and show the throughput of IQ with *i*RRM. Let $\mathbf{R_1}$ be:

$$\mathbf{R_1} = \begin{bmatrix} 0.4 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.1 & 0.3 \\ 0.2 & 0 & 0.5 & 0.3 \end{bmatrix} \tag{5.21}$$

From (5.15) and (5.16), we obtain:

$$\mathbf{R_P}(1) = \begin{bmatrix} 0.3 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.3 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0 & 0.3 \\ 0.2 & 0 & 0.5 & 0.2 \end{bmatrix}$$

$$\mathbf{R_P}(2) = \begin{bmatrix} 0.3 & 0.2 & 0.2 & 0 \\ 0 & 0.3 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0 & 0.3 \\ 0.2 & 0 & 0.3 & 0.2 \end{bmatrix}$$

$$\mathbf{R_P}(3) = \begin{bmatrix} 0.3 & 0.2 & 0 & 0 \\ 0 & 0.3 & 0.2 & 0 \\ 0 & 0.2 & 0 & 0.3 \\ 0.2 & 0 & 0.3 & 0.2 \end{bmatrix}$$

$$\mathbf{R_P}(4) = \begin{bmatrix} 0.3 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0.2 & 0 & 0.1 \\ 0 & 0 & 0.3 & 0.2 \end{bmatrix}$$

$\mathbf{R_P}(4)$ is the matrix of unmatched connections.

From (5.17), we obtain $\mathbf{R_2}$, the traffic matrix at the output ports after N time slots, or the traffic for the matched connections is:

$$\mathbf{R_2} = \begin{bmatrix} 0.1 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.1 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.1 & 0.2 \\ 0.2 & 0 & 0.2 & 0.1 \end{bmatrix} \tag{5.22}$$

Using (5.9), we obtain for throughput by summing the columns of (5.22), and dividing each sum by the sum of the same column of $\mathbf{R_1}$, and summing the results, or:

$$\text{Throughput} = 0.7 + 0.5 + 0.7 + 0.7 = 0.65 \tag{5.23}$$

Therefore, the attained throughput is 65% under this nonuniform traffic and after N time slots.

5.3.3 Load Balanced Birkhoff-von Neumann (LBvN) Switch

We consider a Load Balanced Birkhoff-von Neumann (LBvN) switch [9, 10] in this section. This switch consists of two stages, the first stage load balances the input traffic by using periodic permutations in its configuration, and the second stage, which is a Birkhoff-von Neumann input-queued switch, switches the load-balanced traffic toward the destinations. Figure 5.1(c) shows the architecture of LBvN and the traffic matrix representation of each stage.

Let us denote the traffic coming to the first stage, leaving the first stage, entering the second stage, and leaving the LBvN switch as $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, and $\mathbf{R_4}$, respectively. Here, $\mathbf{R_1}$, $\mathbf{R_2}$, and $\mathbf{R_3}$ are $N \times N$ matrices and $\mathbf{R_4}$ is an $N \times 1$ column vector.

The configuration of the first stage that connects input port, $IP(i)$, to the output of the first stage, or internal output port, $IOP(i)$, at time slot t is represented as an $N \times N$ permutation matrix, $\mathbf{\Pi_{BvN}}(t) = [\pi_{u,v}]$, and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = (u+t) \mod N \\ 0 & \text{elsewhere.} \end{cases} \tag{5.24}$$

The configuration of the first stage can be represented as a compound permutation matrix, $\mathbf{P}$, which is the sum of the permutations used on the first stage over N time slots, as follows:

$$\mathbf{P} = \sum^{N} \mathbf{\Pi_{BvN}}(t) \tag{5.25}$$

Because a switch configuration is repeated every N time slots, the traffic load from the same input going to each VOMQ is $\frac{1}{N}$ of $\mathbf{R_1}$. The traffic matrix $\mathbf{R_2}$ is obtained by using:

$$\mathbf{R_2} = \frac{1}{N}((\mathbf{R_1} * \mathbb{1}) \circ \mathbf{P}) \tag{5.26}$$

Given that

$$\mathbf{P} = \mathbb{1}$$

where $\mathbb{1}$ denotes an $N \times N$ unit matrix. Hence:

$$\mathbf{R_2} = \frac{1}{N}(\mathbf{R_1} * \mathbf{P}) \tag{5.27}$$

Here, $\mathbf{R_2}$ is the aggregate output traffic from the first stage destined to all OPs. This matrix can be further decomposed into N $N \times N$ matrices, $\mathbf{R_2}(v)$, each of which

is the aggregate traffic at the input of the second stage destined to OP_v.

$$\mathbf{R_2} = \sum_{v=0}^{N-1} \mathbf{R_2}(v) \tag{5.28}$$

The second stage of this switch runs a sequence of periodic connection patterns that repeats every N time slots. Hence, the configuration of the second stage can be represented as a compound permutation matrix, $\mathbf{P}$, which is similar to the compound permutation used by the first stage of the switch. Therefore, the traffic forwarded to an OP is:

$$\mathbf{R_3}(v) = (\mathbf{R_2}(v) \circ \mathbf{P}) * \vec{1} \tag{5.29}$$

The traffic forwarded to all OPs, $\mathbf{R_3}$, is:

$$\mathbf{R_3} = [\mathbf{R_3}(0), \cdots, \mathbf{R_3}(N-1)] \tag{5.30}$$

The traffic leaving an OP, $R_4(v)$ is:

$$R_4(v) = (\vec{1})^T * \mathbf{R_3}(v) \tag{5.31}$$

The traffic leaving all OPs, $\mathbf{R_4}$ is:

$$\mathbf{R_4} = [R_4(0), \cdots, R_4(N-1)]^T \tag{5.32}$$

5.3.3.1 Example of a LBvN Switch under Nonuniform Traffic We apply the method on a 4×4 LBvN switch under a nonuniform input matrix to evaluate the throughput of the switch. From Section 5.3.2.1, it is easy to see that LBvN attains 100% throughput under uniform traffic. Let $\mathbf{R_1}$ be the same used in (5.21).

The input traffic into the second stage, $\mathbf{R_2}$, is generated from $\mathbf{R_1}$. From (5.25), the compound permutation matrix of the first stage is:

$$\mathbf{P_1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

This permutation enables connecting each IP to OP and forwarding at a rate of $\frac{1}{N}$, or $\frac{1}{4}$. Using (5.27), we get $\mathbf{R_2}$, the input traffic matrix of the second stage:

$$\mathbf{R_2} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

From (5.28), the traffic matrix at the input of the second stage destined to OPs are:

$$\mathbf{R_2}(0) = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix}$$

$$\mathbf{R_2}(1) = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.025 & 0.025 & 0.025 & 0.025 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R_2}(2) = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.025 & 0.025 & 0.025 & 0.025 \\ 0.125 & 0.125 & 0.125 & 0.125 \end{bmatrix}$$

$$\mathbf{R_2}(3) = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.075 & 0.075 & 0.075 & 0.075 \\ 0.075 & 0.075 & 0.075 & 0.075 \end{bmatrix}$$

The columns of $\mathbf{R_2}(v)$, are the traffic at the queues of the IP of the second stage (IIP), and the rows are the traffic from the IP of the first stage. Using (5.29), the traffic forwarded into the second stage destined for each OP, $\mathbf{R_3}(v)$, is:

$$\mathbf{R_3}(0) = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{R_3}(1) = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ 0 \end{bmatrix}$$

$$\mathbf{R_3}(2) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.5 \end{bmatrix}$$

$$\mathbf{R_3}(3) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.3 \\ 0.3 \end{bmatrix}$$

The rows of $\mathbf{R_3}(v)$ represent the traffic from IPs forwarded through an IIP. Using (5.30), the aggregate traffic forwarded into the second stage, $\mathbf{R_3}$, is:

$$\mathbf{R_3} = \begin{bmatrix} 0.4 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.1 & 0.3 \\ 0.2 & 0 & 0.5 & 0.3 \end{bmatrix}$$

The traffic leaving each OP is obtained from (5.31):

$$R_4(0) = 1 \tag{5.33}$$

$$R_4(1) = 1 \tag{5.34}$$

$$R_4(2) = 1 \tag{5.35}$$

$$R_4(3) = 1 \tag{5.36}$$

The traffic leaving alls OPs is obtained from (5.32):

$$\mathbf{R_4} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{5.37}$$

Equation (5.37) is the output traffic and by using (5.9), it is easy to see that this switch achieves 100% throughput under nonuniform traffic. These results are consistent with those presented in [9, 10].

5.3.4 Non-blocking Memory-Memory-Memory with Extended Memory (MM^{e}M) Clos-Network Switch

Now let us focus on a non-blocking memory-memory-memory with extended memory Clos-network switch (MM^{e}M) [23, 26, 27]. This switch is a three-stage buffered Clos-network architecture consisting of k $n \times m$ Input modules (IMs), k $m \times n$ Output modules (OMs), and m $p \times p$ Central modules (CMs). There are nN crosspoint buffers in an IM and OM, and kN crosspoint buffers in a CM. Each switch module has per-output flow queues to avoid HoL blocking. Round-robin or longest queue first arbitrations can be used in the IM, while round-robin is used in the CMs and OMs. The switch uses round-robin to arbitrate which queue forwards a cell to the output links of the modules of all three stages. Let us denote the traffic coming to the IPs, CMs, OPs, and the traffic leaving MM^{e}M as $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, and R_4 respectively. Here, $\mathbf{R_1}$, $\mathbf{R_2}$, and $\mathbf{R_3}$ are $N \times N$ matrices and $\mathbf{R_4}$ is an $N \times 1$ column vector.

Round-robin arbitration is used in IMs to select $IP(i, s)$ permitted to forward a cell to the to the virtual central module queue ($VCMQ(i, s, r)$) at $IM(i)$ that stores cells from $IP(i, s)$ going through $CM(r)$ to its output port. This arbitration scheme enables forwarding a cell from $IP(i, s)$ to the queues in IM, and in turn forwarding cells from $IM(i)$ to output link $L_I(i, r)$ that connects the $IM(i)$ to $CM(r)$. This interconnection occurs once every n time slots for a fully loaded switch. Similar to the selection of queues in IM, round-robin arbitration is used at a CM to select the per-output flow queue ($POFQ(i, r, j, d)$) that stores cells from $L_I(i, r)$ and destined to $OP(j, d)$. Figure 5.1(d) shows the architecture of MM^{e}M.

The specific configurations of the IM and CM are as follows. At time slot t, $IP(i,s)$ sends a cell to $VCMQ(i,s,r)$, and this VCMQ is connected to $L_I(i,r)$ in the following time slot, as follows:

$$r = (s+t) \mod m \tag{5.38}$$

Each CM input $L_I(i,r)$ is connected to $POFQ(i,r,j,d)$ and then this queue is connected to $L_C(r,j)$ as follows:

$$j = (r+t) \mod k \tag{5.39}$$

Then $\mathbf{R_2}$ is the traffic forwarded to the CMs and the product of having $\mathbf{R_1}$ switched by the permutations used in the configurations of IMs. The configuration of the IM stage at time slot t that interconnects $IP(i,s)$ to $VCMQ(i,r,j)$, and in turn interconnects to $L_I(i,r)$, are represented as an $N \times N$ permutation matrix, $\mathbf{\Pi_{Clos}}(t) = [\pi_{u,v}]$, where r is determined from (5.38) and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = rk + i \\ 0 & \text{otherwise.} \end{cases} \tag{5.40}$$

The configuration of the IM stage can be represented as a compound permutation matrix, $\mathbf{P_1}$, which is the sum of the IM permutations over m time slots as follows,

$$\mathbf{P_1} = \sum^{m} \mathbf{\Pi_{Clos}}(t) \tag{5.41}$$

Because the configuration repeats every m time slots, the traffic load from the same input going to each VCMQ is $\frac{1}{m}$ of the traffic load of $\mathbf{R_1}$. The traffic matrix $\mathbf{R_2}$ is obtained using:

$$\mathbf{R_2} = \frac{1}{m}((\mathbf{R_1} * \mathbb{1}) \circ \mathbf{P_1}) \tag{5.42}$$

Here, $\mathbf{R_2}$ is the aggregate traffic being forwarded to CMs destined to all OPs. This matrix can be further decomposed into N $N \times N$ submatrices, $\mathbf{R_2}(j,d)$, each of which is the aggregate traffic at CMs destined to $OP(j,d)$.

$$\mathbf{R_2} = \sum_{j=0}^{j=k-1} \sum_{d=0}^{d=n-1} \mathbf{R_2}(j,d) \tag{5.43}$$

The configuration of the CM stage at time slot t that connects $I_c(r,p)$ to $L_{C(r,j)}$ may be represented as an $N \times N$ permutation matrix, $\mathbf{\Phi}(t) = [\phi_{u,v}]$, where j is determined from (5.39) and the matrix element:

$$\phi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = jk + r \\ 0 & \text{otherwise.} \end{cases} \tag{5.44}$$

Similarly, the switching process at the CM stage is represented by a compound permutation matrix $\mathbf{P_2}$, which is the sum of p permutations of the CM stage over p time slots. Here,

$$\mathbf{P_2} = \sum^{p} \mathbf{\Phi}(t) \tag{5.45}$$

The traffic forwarded to the OMs and queued at the CB of an OP, $\mathbf{R_3}(j,d)$ is:

$$\mathbf{R_3}(j,d) = (\mathbf{R_2}(j,d) \circ \mathbf{P_2}) * \vec{1} \tag{5.46}$$

The aggregate traffic forwarded to the OMs, $\mathbf{R_3}$ is:

$$\mathbf{R_3} = [\mathbf{R_3}(0,0), \cdots, \mathbf{R_3}(k-1, n-1)] \tag{5.47}$$

The traffic leaving an OP, $R_4(j,d)$, is:

$$R_4(j,d) = (\vec{1})^T * \mathbf{R_3}(j,d) \tag{5.48}$$

The aggregate traffic leaving all OPs, $\mathbf{R_4}$, is:

$$\mathbf{R_4} = [R_4(0,0), \cdots, R_4(p-1, n-1)]^T \tag{5.49}$$

5.3.4.1 Example of a Buffered Clos-network Switch with Extended Memory (MM^{e}M) under Nonuniform Traffic In this example, we consider a nonuniform input matrix as the traffic coming into MM^{e}M and calculate the switch throughput. We only consider nonuniform traffic in this example because from Section 5.3.2.1, it is easy to see that MM^{e}M attains 100% throughput under uniform traffic. Let $\mathbf{R_1}$ be the that used in (5.21).

From (5.41), the compound permutation matrix of the IM is:

$$\mathbf{P_1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

This permutation enables connecting each IP to the output of the IM. Using (5.42), we get:

$$\mathbf{R_2} = 1/2 \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

From (5.43), the traffic matrix at the CMs destined for the different OPs are:

$$\mathbf{R_2}(0,0) = \begin{bmatrix} 0.2 & 0 & 0.2 & 0 \\ 0.1 & 0 & 0.1 & 0 \\ 0 & 0.1 & 0 & 0.1 \\ 0 & 0.1 & 0 & 0.1 \end{bmatrix}$$

$$\mathbf{R_2}(0,1) = \begin{bmatrix} 0.1 & 0 & 0.1 & 0 \\ 0.2 & 0 & 0.2 & 0 \\ 0 & 0.2 & 0 & 0.2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R_2}(1,0) = \begin{bmatrix} 0.1 & 0 & 0.1 & 0 \\ 0.1 & 0 & 0.1 & 0 \\ 0 & 0.05 & 0 & 0.05 \\ 0 & 0.25 & 0 & 0.25 \end{bmatrix}$$

$$\mathbf{R_2}(1,1) = \begin{bmatrix} 0.1 & 0 & 0.1 & 0 \\ 0.1 & 0 & 0.1 & 0 \\ 0 & 0.15 & 0 & 0.15 \\ 0 & 0.15 & 0 & 0.15 \end{bmatrix}$$

From (5.45), the compound permutation matrix for the CM stage for this switch is:

$$\mathbf{P_2} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

From (5.46), the traffic matrix forwarded to OMs and queued at the CBs of each OP, $\mathbf{R_3}(0,0)$ to $\mathbf{R_3}(1,1)$ are derived:

$$\mathbf{R_3}(0,0) = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{R_3}(0,1) = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ 0 \end{bmatrix}$$

$$\mathbf{R_3}(1,0) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.5 \end{bmatrix}$$

$$\mathbf{R_3}(1,1) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.3 \\ 0.3 \end{bmatrix}$$

From (5.47), the traffic matrix forwarded to OMs and queued at the CBs of all OPs, $\mathbf{R_3}$ is:

$$\mathbf{R_3} = \begin{bmatrix} 0.4 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.1 & 0.3 \\ 0.2 & 0 & 0.5 & 0.3 \end{bmatrix}$$

Using (5.48), we obtain the traffic leaving each OP, or:

$$R_4(0) = 1 \tag{5.50}$$

$$R_4(1) = 1 \tag{5.51}$$

$$R_4(2) = 1 \tag{5.52}$$

$$R_4(3) = 1 \tag{5.53}$$

Using (5.49), we obtain the traffic leaving all OPs, or:

$$\mathbf{R_4} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{5.54}$$

Equation (5.54) is the output traffic and using (5.9) shows that this switch achieves 100% throughput under non-uniform traffic. These results are consistent with those presented in the original paper [23, 26, 27].

5.4 Conclusions

We have shown in this chapter that the operation of a switch affects the average traffic coming into the switch and the overall performance of the switch. These operations can be represented as matrix operations, which may be used to estimate the throughput of a switch. We have shown the application of the proposed approach on examples of switch architectures with single or multiple stages, and with different queueing strategies and their corresponding configuration schemes. The results obtained in the presented examples are consistent with the known performance of such switches. By the examples provided, the tool is shown to be practical and can be used to analyze the performance of a wide set of packet switches.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Conclusion

This work addressed low performance, scalability, and high configuration complexity in large scale Clos-network packet switches. We introduced three high performance switches: A split-central-buffered Load-Balancing Clos-network (LBC), ThRee-stage Clos swItch with queues at the middle stage and DEtermiNisTic scheduling (TRIDENT), and Load-Balancing Clos-Network Switch with Virtual Input-Module Output Queues at Central Stage (VINE), their configuration schemes and in-sequence mechanisms.

The results obtained from analysis and computer simulations show that for any admissible independent and identically distributed traffic, these switches achieve 100% throughput. Unlike other existing switching architectures, LBC, TRIDENT, and VINE achieve high performance, configuration simplicity, and in-sequence service without memory speedup and central module expansion. Also VINE emulates the performance of an ideal OQ switch.

Future work

This work analyzed the performance of LBC, TRIDENT, and VINE under unicast traffic. Future analysis under multicast traffic can be performed to ascertain the suitability of these switches for networks with multicast traffic.

The switching architectures and their configuration schemes, in-sequence mechanisms, and flow control mechanism introduced in this work can be adopted for packet networks, such as datacenter networks.

In this work, we introduced the use of matrix analysis as a tool for throughput calculation of packet switches and an extension of this tool was introduced in VINE for

stability analysis. Future works can be performed on applying this tool to calculate throughput in packet networks, such as datacenter networks. Matrix analysis can also be adopted for analyzing other performance metrics.

BIBLIOGRAPHY

[1] T.E. Anderson, S.S. Owicki, J.B. Saxe, and C.P. Thacker. High speed switch scheduling for local area networks. In *ACM SIGPLAN Notices*, volume 27, pages 98–110. ACM, 1992.

[2] T.E. Anderson, S.S. Owicki, J.B. Saxe, and C.P. Thacker. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems (TOCS)*, 11(4):319–352, 1993.

[3] G. Bastin, B. Haut, J.M. Coron, and B. d Andrea-Novel. Lyapunov stability analysis of networks of scalar conservation laws. *Networks and Heterogeneous media*, 2(4):749, 2007.

[4] D.P. Bertsekas, R.G. Gallager, and P. Humblet. *Data networks*, volume 2. Prentice-Hall International New Jersey, 1992.

[5] G. Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumán Rev. Ser. A*, 5:147–151, 1946.

[6] L. Cai, R. Rojas-Cessa, and T. Kijkanjanarat. Avoiding speedup from bandwidth overhead in a practical output-queued packet switch. In *IEEE International Conference on Communications*, pages 1–5, 2011.

[7] C.S. Chang, W. Chen, and H.Y. Huang. On service guarantees for input-buffered crossbar switches: a capacity decomposition approach by birkhoff and von neumann. In *Quality of Service (IWQoS). Seventh International Workshop on*, pages 79–86. IEEE, 1999.

[8] C.S. Chang, W. Chen, and H.Y. Huang. Birkhoff-von neumann input buffered crossbar switches. In *IEEE INFOCOM. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1614–1623, 2000.

[9] C.S. Chang, D.S. Lee, and Y.S. Jou. Load balanced Birkhoff–von Neumann switches, part i: one-stage buffering. *Computer Communications*, 25(6):611–622, 2002.

[10] C.S. Chang, D.S. Lee, and C.M. Lien. Load balanced Birkhoff-von Neumann switches, part ii: Multi-stage buffering. *Computer Communications*, 25(6):623–634, 2002.

[11] C.H. Chao and Y. Hsu. Improved static desynchronization scheme for clos-network switches. In *Computing Technology and Information Management (ICCM), 8th International Conference on*, volume 2, pages 659–664. IEEE, 2012.

[12] H.J. Chao, Z. Jing, and S.Y. Liew. Matching algorithms for three-stage bufferless Clos network switches. *Communications Magazine, IEEE*, 41(10):46–54, 2003.

[13] H.J. Chao, J. Park, S. Artan, S. Jiang, and G. Zhang. Trueway: a highly scalable multi-plane multi-stage buffered packet switch. In *IEEE High Performance Switching and Routing, Workshop on*, pages 246–253, 2005.

[14] A. Charny, P. Krishna, N. Patel, and R. Simcoe. Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup. In *Quality of Service (IWQoS). Sixth International Workshop on*, pages 235–244. IEEE, 1998.

[15] D.X. Chen and J.W. Mark. Scoq: A fast packet switch with shared concentration and output queueing. *IEEE/ACM Transactions on Networking (TON)*, 1(1):142–151, 1993.

[16] F.M. Chiussi, J.G. Kneuer, and V.P. Kumar. Low-cost scalable switching solutions for broadband networking: the ATLANTA architecture and chipset. *IEEE Communications Magazine*, 35(12):44–53, 1997.

[17] N. Chrysos and M. Katevenis. Scheduling in non-blocking buffered three-stage switching fabrics. In *INFOCOM*, volume 6, pages 1–13, 2006.

[18] S.T. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input/output-queued switch. *IEEE Journal on Selected Areas in Communications*, 17(6):1030–1039, 1999.

[19] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(2):406–424, 1953.

[20] J.G. Dai and B. Prabhakar. The throughput of data switches with and without speedup. In *IEEE INFOCOM. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 556–564. IEEE, 2000.

[21] E. Del Re and R. Fantacci. Performance evaluation of input and output queueing techniques in atm switching systems. *IEEE Transactions on communications*, 41(10):1565–1575, 1993.

[22] A. Dixit, P. Prakash, Y.C. Hu, and R.R. Kompella. On the impact of packet spraying in data center networks. In *IEEE INFOCOM*, pages 2130–2138, 2013.

[23] Z. Dong and R. Rojas-Cessa. Non-blocking memory-memory-memory Clos-network packet switch. In *IEEE 34th Sarnoff Symposium*, pages 1–5, 2011.

[24] Z. Dong and R. Rojas-Cessa. MCS: buffered Clos-network switch with in-sequence packet forwarding. In *IEEE 35th Sarnoff Symposium*, pages 1–6, 2012.

[25] Z. Dong and R. Rojas-Cessa. Throughput analysis of shared-memory crosspoint buffered packet switches. *IET communications*, 6(9):1045–1053, 2012.

[26] Z. Dong, R. Rojas-Cessa, and E. Oki. Buffered clos-network packet switch with per-output flow queues. *Electronics letters*, 47(1):32–34, 2011.

[27] Z. Dong, R. Rojas-Cessa, and E. Oki. Memory-memory-memory Clos-network packet switches with in-sequence service. In *IEEE High Performance Switching and Routing (HPSR), 12th International Conference on*, pages 121–125, 2011.

[28] Y. Gao, Z. Qiu, M. Zhang, and Y. Jiang. Distributed weight matching dispatching scheme in msm clos-network packet switches. *IEEE Communications Letters*, 17(3):580–583, 2013.

[29] F. Hassen and L. Mhamdi. High-capacity Clos-network switch for data center networks. In *IEEE International Conference on Communications*, pages 1–7, 2017.

[30] M.G. Hluchyj and M.J. Karol. Queueing in high-performance packet switching. *IEEE Journal on selected Areas in Communications*, 6(9):1587–1597, 1988.

[31] B. Hu and K.L. Yeung. On joint sequence design for feedback-based two-stage switch architecture. In *IEEE High Performance Switching and Routing. International Conference on*, pages 110–115, 2008.

[32] B. Hu, K.L. Yeung, Q. Zhou, and C. He. On iterative scheduling for input-queued switches with a speedup of $2 - 1/n$. *IEEE/ACM Transactions on Networking*, 24(6):3565–3577, 2016.

[33] Y. Hu, M. Li, Q. Zhou, J. Li, Y. Zhang, and J. Li. A module matched first dispatching algorithm for msm clos-network switches. In *IEEE International Conference on Electronics, Communications and Control (ICECC)*, pages 4086–4089, 2011.

[34] I. Iliadis and W.E. Denzel. Performance of packet switches with input and output queueing. In *IEEE Communications, Including Supercomm Technical Sessions. (SUPERCOMM/ICC). International Conference on*, pages 747–753, 1990.

[35] S.M. Irteza, H.M. Bashir, T. Anwar, I.A. Qazi, and F.R. Dogar. Load balancing over symmetric virtual topologies. In *INFOCOM - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[36] Y.C. Jenq. Performance analysis of a packet switch based on single-buffered banyan network. *IEEE Journal on Selected Areas in Communications*, 1(6):1014–1021, 1983.

[37] Y. Kanizo, D. Hay, and I. Keslassy. The crosspoint-queued switch. In *IEEE INFOCOM*, pages 729–737, 2009.

[38] M. Karol, M. Hluchyj, and S. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, 35(12):1347–1356, 1987.

[39] I. Keslassy and N. McKeown. Maintaining packet order in two-stage switches. In *IEEE INFOCOM. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1032–1041, 2002.

[40] J. Kleban and H. Santos. Packet dispatching algorithms with the static connection patterns scheme for three-stage buffered clos-network switches. In *IEEE International Conference on Communications*, pages 6319–6324, 2007.

[41] J. Kleban, M. Sobieraj, and S. Weclewski. The modified MSM Clos switching fabric with efficient packet dispatching scheme. In *IEEE High Performance Switching and Routing (HPSR), Workshop on*, pages 1–6, 2007.

[42] J. Kleban and U. Suszynska. Static dispatching with internal backpressure scheme for SMM Clos-network switches. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 000654–000658, 2013.

[43] H.Y. Lee, F.K. Hwang, and J.D. Carpinelli. A new decomposition algorithm for rearrangeable clos interconnection networks. *IEEE Transactions on Communications*, 44(11):1572–1578, 1996.

[44] T.T. Lee and C.H. Lam. Path switching-a quasi-static routing scheme for large-scale ATM packet switches. *IEEE Journal on Selected Areas in Communications*, 15(5):914–924, 1997.

[45] J. Li and N. Ansari. Enhanced birkhoff-von neumann decomposition algorithm for input queued switches. *IEE Proceedings-Communications*, 148(6):339–342, 2001.

[46] X. Li, Z. Zhou, and M. Hamdi. Space-memory-memory architecture for clos-network packet switches. In *IEEE International Conference on Communications*, volume 2, pages 1031–1035.

[47] C.B. Lin and R. Rojas-Cessa. Module-first matching schemes for scalable input-queued space-space-space clos-network packet switches. In *IEEE International Conference on Communications (ICC)*, pages 5669–5673, 2008.

[48] C.B. Lin and R. Rojas-Cessa. Minimizing scheduling complexity with a Clos-network space-space-memory (SSM) packet switch. In *IEEE High Performance Switching and Routing (HPSR), 14th International Conference on*, pages 15–20, 2013.

[49] M. Lin and N. McKeown. The throughput of a buffered crossbar switch. *IEEE Communications Letters*, 9(5):465–467, 2005.

[50] K. Liu, J. Yan, and J. Lu. Fault-tolerant cell dispatching for onboard space-memory-memory clos-network packet switches. In *IEEE High Performance Switching and Routing (HPSR), 16th International Conference on*, pages 1–6, 2015.

[51] F.A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and E. Neri. Packet scheduling in input-queued cell-based switches. In *IEEE INFOCOM. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1085–1094, 2001.

[52] N. McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM transactions on networking*, 7(2):188–201, 1999.

[53] N. McKeown, B. Prabhakar, and M. Zhu. Matching output queueing with combined input and output queueing. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 35, pages 595–603. Citeseer, 1997.

[54] N. McKeown, P. Varaiya, and J. Walrand. Scheduling cells in an input-queued switch. *Electronics Letters*, 29(25):2174–2175, 1993.

[55] N.W. McKeown. *Scheduling algorithms for input-queued cell switches*. PhD thesis, University of California, Berkeley, 1995.

[56] A. Mekkittikul and N. McKeown. A starvation-free algorithm for achieving 100% throughput in an input-queued switch. In *Proc. of the IEEE International Conference on Communication Networks*, volume 96, pages 226–231. Citeseer, 1996.

[57] A. Mekkittikul and N. McKeown. Scheduling voq switches under non-uniform traffic. *CSL Technical Report, CSL-TR 97-747*, 1997.

[58] A. Mekkittikul and N. McKeown. A practical scheduling algorithm to achieve 100% throughput in input-queued switches. In *IEEE INFOCOM. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 792–799, 1998.

[59] L. Mhamdi and M. Hamdi. Cbf: A high-performance scheduling algorithm for buffered crossbar switches. In *IEEE High Performance Switching and Routing (HPSR). Workshop on*, pages 67–72.

[60] E. Oki, N. Kitsuwan, and R. Rojas-Cessa. Analysis of space-space-space Clos-network packet switch. In *IEEE Computer Communications and Networks (ICCCN). 18th International Conference on*, pages 1–6, 2009.

[61] A. Pattavina and G. Bruzzi. Analysis of input and output queueing for nonblocking atm switches. *IEEE/ACM Transactions on Networking (TON)*, 1(3):314–328, 1993.

[62] B. Prabhakar and N. McKeown. On the speedup required for combined input-and output-queued switching. *Automatica*, 35(12):1909–1920, 1999.

[63] Konghong Pun and Mounir Hamdi. Static round-robin dispatching schemes for clos-network switches. In *IEEE High Performance Switching and Routing. Merging Optical and IP Technologies. Workshop on*, pages 329–333, 2002.

[64] R. Rojas-Cessa. *Interconnections for Computer Communications and Packet Networks*. CRC Press, 2016.

[65] R. Rojas-Cessa and Z. Dong. Combined input-crosspoint buffered packet switch with flexible access to crosspoints buffers. In *Devices, Circuits and Systems, Proceedings of the 6th International Caribbean Conference on*, pages 255–260. IEEE, 2006.

[66] R. Rojas-Cessa and Z. Dong. Load-balanced combined input-crosspoint buffered packet switches. *IEEE Transactions on Communications*, 59(5):1421–1433, 2011.

[67] R. Rojas-Cessa, Z. Guo, and N. Ansari. Combining distributed and centralized arbitration schemes for combined input-crosspoint buffered packet switches. In *Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005 13th IEEE International Conference on*, volume 2, pages 766–770.

[68] R. Rojas-Cessa, Z. Guo, and N. Ansari. On the maximum throughput of a combined input-crosspoint queued packet switch. *IEICE Transactions on Communications*, 89(11):3120–3123, 2006.

[69] R. Rojas-Cessa and C.B. Lin. Matching schemes with captured-frame eligibility for input-queued packet switches. In *IEEE International Conference on Communications*, volume 2, pages 972–976. IEEE, 2005.

[70] R. Rojas-Cessa and C.B. Lin. Scalable two-stage Clos-network switch and module-first matching. In *IEEE High Performance Switching and Routing, Workshop on*, pages 6–pp, 2006.

[71] R. Rojas-Cessa, C.B. Lin, and Z. Dong. Space-space-memory (ssm) clos-network packet switch, March 31 2015. US Patent 8,995,456.

[72] R. Rojas-Cessa, E. Oki, and H.J. Chao. CIXOB-k: Combined input-crosspoint-output buffered packet switch. In *IEEE Global Telecommunications Conference (GLOBECOM)*, volume 4, pages 2654–2660, 2001.

[73] R. Rojas-Cessa, E. Oki, and H.J. Chao. Maximum weight matching dispatching scheme in buffered Clos-network packet switches. In *IEEE International Conference on Communications*, volume 2, pages 1075–1079, 2004.

[74] R. Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao. CIXB-1: combined input-one-cell-crosspoint buffered switch. In *IEEE High Performance Switching and Routing, Workshop on*, pages 324–329, 2001.

[75] D. Shah and M. Kopikare. Delay bounds for approximate maximum weight matching algorithms for input queued switches. In *IEEE INFOCOM. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1024–1031, 2002.

[76] L. Shi, B. Liu, C. Sun, Z. Yin, L.N. Bhuyan, and H.J. Chao. Load-balancing multipath switching system with flow slice. *IEEE Transactions on Computers*, 61(3):350–365, March 2012.

[77] D.C. Stephens and H. Zhang. Implementing distributed packet fair queueing in a scalable switch architecture. In *IEEE INFOCOM. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 282–290, 1998.

[78] Y. Tian, X. Zhang, M. Li, and H. Zhang. An in-sequence guaranteed space-memory-memory clos-network architecture. In *ICT Convergence (ICTC), International Conference on*, pages 354–359. IEEE, 2012.

[79] J. Von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, 2:5–12, 1953.

[80] Y. Xia and H.J. Chao. Module-level matching algorithms for msm clos-network switches. In *IEEE High Performance Switching and Routing (HPSR), 13th International Conference on*, pages 36–43, 2012.

[81] Y. Xia and H.J. Chao. On practical stable packet scheduling for bufferless three-stage clos-network switches. In *IEEE High Performance Switching and Routing (HPSR), 14th International Conference on*, pages 7–14, 2013.

[82] Y. Xia, M. Hamdi, and H.J. Chao. A practical large-capacity three-stage buffered Clos-network switch architecture. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):317–328, 2016.

[83] H. Yu, S. Ruepp, and M.S. Berger. Out-of-sequence prevention for multicast input-queuing space-memory-memory clos-network. *IEEE Communications Letters*, 15(7):761–763, 2011.

[84] M. Zhang, Z. Qiu, and Y. Gao. Space-memory-memory Clos-network switches with in-sequence service. *Communications, IET*, 8(16):2825–2833, 2014.